CMP
United Business Media

REVIEWS: 3DBOXX & DELL M50 WORKSTATIONS, MORE

JULY 2002

# Game Developer

ON THE FRONT LINE OF GAME [illegible]ION

## Facing the Future of Game AI

## State Of The Industry Report

## Our 2nd Annual Salary Survey

How Do You Measure Up?

## Surreal's DRAKAN: THE ANCIENTS' GATES

When A Sequel Takes On A Life Of Its Own

WWW.GDMAG.COM
$5.95US $6.95CAN
07
0 71486 02133 9
Display Until July 26, 2002

# Buy an engine. Gain a partner.

When you license NDL's NetImmerse 3D game engine and development toolkit, you get much more than great code. You gain a development partner with engineers dedicated to doing everything possible to ensure the success of your title.

NetImmerse is the only engine that can deliver all of this:

- A complete toolset that accelerates your art pipeline.
- Optimized performance across all platforms.
- Full source code for all libraries and tools.
- Responsive support from those who develop the engine.

**Great software. Solid partner. Proven success.**

## NetImmerse

***The fastest path from conception to completion***

**925.556.9995 • sales@ndl.com**

ndl

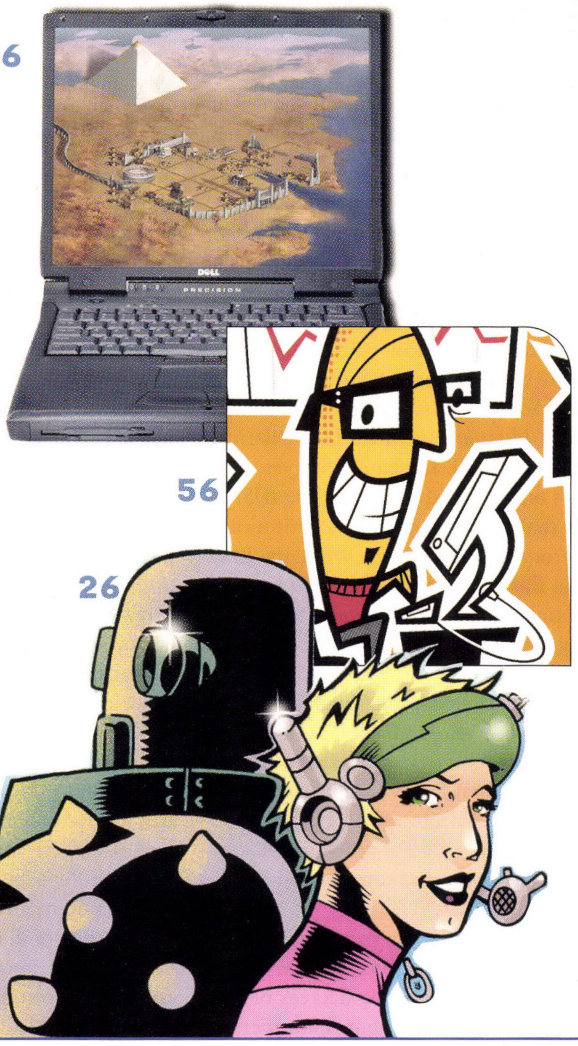
*The Elder Scrolls III: Morrowind courtesy of Bethesda Softworks.*
*Star Trek® Bridge Commander™ courtesy Paramount Pictures. ™, ®, & © 2002 Par. Pic. All Rights Reserved.*
*Dark Age of Camelot courtesy of Mythic Games.*
*Tennis Master Series courtesy of Microids.*
*Freedom Force courtesy of Irrational Games.*

**www.ndl.com**

# CONTENTS

VOLUME 9, NUMBER 7

JULY 2002




## FEATURES

**26 Game AI: The State of the Industry, 2001-2002**
It's that time of year again! As Woodcock delves into the present state of AI in game development, find out where you, your company, and your peers stand, and where the industry's going.
*Steven Woodcock*

**32 *Game Developer*'s Second Annual Salary Survey**
The past year brought a lot of changes to the game industry and the economy. See how you're doing compared to the 2,524 game developers represented in this year's salary survey.
*Jennifer Olsen*

**40 Postmortem: Surreal's DRAKAN: THE ANCIENTS' GATES**
After listening to players, Surreal took what started out as a simple sequel for a 1999 PC title and metamorphosed it into a Playstation game that took the strengths of its predecessor and, through days and nights of redesigning and refocusing, became a major new title in its own right.
*Richard Rouse III*

## DEPARTMENTS

2 **GAME PLAN** *Jennifer Olsen*
So It's Come to This

4 **INDUSTRY WATCH** *Daniel Huebner*
Xbox flounders, Interplay stays afloat, EA lounges by the pool

6 **PRODUCT REVIEWS**
3DBOXX and Dell's M50 Workstations, plus *AI Game Programming Wisdom*

12 **PROFILES** *Jennifer Olsen*
Revelations from id's Graeme Devine

## COLUMNS

14 **THE INNER PRODUCT** *Jonathan Blow*
Transmitting Vectors

18 **ARTIST'S VIEW** *Hayden Duvall*
Textures: From Source to Screen, Part 3

22 **SOUND PRINCIPLES** *Andrew Boyd*
You Get What You Ask For

24 **BETTER BY DESIGN** *Noah Falstein*
Turn-offs

56 **SOAPBOX** *Bruce Shelley*
Sells Like Team Spirit

**COVER** *for this issue of* Game Developer *magazine created exclusively by the fine folks at Surreal.*

*Game Developer* (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian GST: 124513185. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence.

# GAME PLAN

LETTER FROM THE EDITOR

## So It's Come to This

The month leading up to E3 this year was marred by a couple of unfortunate public policy incidents involving the sale of videogames and the constitutional rights of game developers and publishers. By now you are likely familiar with the case of *Interactive Digital Software Association v. St. Louis County, Missouri*, in which U.S. District Judge Stephen N. Limbaugh Sr. (yes, he's Rush's uncle) failed to uphold the IDSA's claim that games constitute constitutionally protected free speech. While the findings in this case are baffling at best and potentially disastrous at worst, freedom-loving developers can only wait this one out on appeal.

Just days after the Limbaugh decision came down, Rep. Joe Baca (R-Calif.) and 21 co-sponsors introduced HR 4652, the Protect Children From Video Game Sex and Violence Act of 2002, a bill that would make it a federal crime to sell certain games to minors, complete with jail time for scofflaws.

While I always enjoy receiving mail from readers, as my inbox flooded with e-mails expressing varying degrees of outrage, I couldn't help but think that the recreational pundits in our industry spend entirely too much time preaching to the choir and virtually no time trying to convert the unclean masses. I'm referring — not literally, of course — to our elected officials.

So I am hereby posing a challenge to every single one of you: You must write your congressman.

No, I'm not kidding. You know that bills like HR 4652 get introduced because members of Congress have constituencies back home that expect certain things from them. It's hard for me to imagine Rep. Baca not receiving at least one letter in his past three years in office from a distraught constituent asking, "Joe, what are you doing to protect our children?" And the ones he doesn't hear from? Well, he can only guess what's on their mind, but he probably won't.

So while this bill languishes in committee, take the time to mail (yes, dead tree and stamp) your congressman a letter. Introduce yourself. Explain a bit about what your job is, why you do it, how your company benefits the local economy and tax base. Perhaps modestly point out that the industry in which you work added the rough equivalent of the gross domestic product of Panama to the U.S. economy last year and helps drive other major high-tech sectors. You can let them know that the ESRB has an industry-wide rating system that the FTC seems to think works pretty well, and that criminalizing business owners is both extreme and unnecessary. Use this bill as a backdrop for simply opening a dialogue.

You're too busy, you say? Nonsense. E3 is over and Christmas is still a ways off. Now if you're just plain lazy, that's O.K. I have taken the liberty of posting my own letter to my congressperson at www.gdmag.com/congress.htm that you can use for some inspiration if you wish, and also some letter-writing tips. I've also posted a form letter that you can just outright copy, although be warned that form letters (and they're easily spotted) carry far less weight at your legislator's offices. So do something to personalize it if at all possible.

Every single one of you who is a registered voter can find at least five minutes to do this. Do it not just for yourself but for your colleagues and your collective future. Once you've sent your letter, e-mail me and let me know, and I'll try to offer an update in a future issue of how we're doing.

Lawmakers reckon that each letter they receive in the mail represents the viewpoint of roughly 500 constituents. If you're tired of negative, one-sided PR against the game industry and developers, then put some good PR for us out there to people whose opinions matter and to whom your opinion matters.

When you're done, you can celebrate by playing THE RESIDENT OF EVIL CREEK.

Jennifer

Jennifer Olsen
Editor-In-Chief


**GameDeveloper**

600 Harrison Street, San Francisco, CA 94107 t: 415.947.6000 f: 415.947.6090

**Publisher**
Jennifer Pahlka *jpahlka@cmp.com*

**EDITORIAL**

**Editor-In-Chief**
Jennifer Olsen *jolsen@cmp.com*

**Managing Editor**
Everard Strong *estrong@cmp.com*

**Production Editor**
Olga Zundel *ozundel@cmp.com*

**Product Review Editor**
Daniel Huebner *dan@gamasutra.com*

**Art Director**
Elizabeth von Büdingen *evonbudingen@cmp.com*

**Editor-At-Large**
Chris Hecker *checker@d6.com*

**Contributing Editors**
Jonathan Blow *jon@bolt-action.com*
Hayden Duvall *hayden@confounding-factor.com*
Noah Falstein *noah@theinspiracy.com*

**Advisory Board**
Hal Barwood LucasArts
Ellen Guon Beeman Beemania
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Ion Storm
Paul Steed WildTangent

**ADVERTISING SALES**

**Director of Sales & Marketing**
Greg Kerwin e: *gkerwin@cmp.com* t: 415.947.6218

**National Sales Manager**
Jennifer Orvik e: *jorvik@cmp.com* t: 415.947.6217

**Senior Account Manager, Eastern Region & Europe**
Afton Thatcher e: *athatcher@cmp.com* t: 415.947.6224

**Account Manager, Northern California & Southeast**
Susan Kirby e: *skirby@cmp.com* t: 415.947.6226

**Account Manager, Recruitment**
Raelene Maiben e: *rmaiben@cmp.com* t: 415.947.6225

**Account Manager, Western Region & Asia**
Craig Perreault e: *cperreault@cmp.com* t: 415.947.6223

**Account Representative**
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227

**ADVERTISING PRODUCTION**

**Vice President, Manufacturing** Bill Amstutz
**Advertising Production Coordinator** Kevin Chanel
**Reprints** Cindy Zauss t: 909.698.1780

**GAMA NETWORK MARKETING**

**Senior MarCom Manager** Jennifer McLean
**Marketing Coordinator** Scott Lyon

**CIRCULATION**

BPA INTERNATIONAL
*Game Developer* is BPA approved

**Group Circulation Director** Catherine Flynn
**Circulation Manager** Ron Escobar
**Circulation Assistant** Ian Hay
**Newsstand Analyst** Pam Santoro

**SUBSCRIPTION SERVICES**

**For information, order questions, and address changes**
t: 800.250.2429 or 847.647.5928 f: 847.647.5972
e: *gamedeveloper@halldata.com*

**INTERNATIONAL LICENSING INFORMATION**

**Mario Salinas**
t: 650.513.4234 f: 650.513.4482 e: *msalinas@cmp.com*

**CMP MEDIA MANAGEMENT**

**President & CEO** Gary Marshall
**Executive Vice President & CFO** John Day
**Chief Operating Officer** Steve Weitzner
**Chief Information Officer** Mike Mikos
**President, Technology Solutions Group** Robert Faletra
**President, Business Technology Group** Adam K. Marder
**President, Healthcare Group** Vicki Masseria
**President, Electronics Group** Jeff Patterson
**President, Specialized Technologies Group** Regina Starr Ridley
**Senior Vice President, Global Sales & Marketing** Bill Howard
**Senior Vice President, HR & Communications** Leah Landro
**Vice President & General Counsel** Sandra Grayson

CMP United Business Media

GamaNetwork

WWW.GAMANETWORK.COM

QUALCOMM

# RULES – DISENGAGED.

## IMAGINATION – FULL STEAM AHEAD.

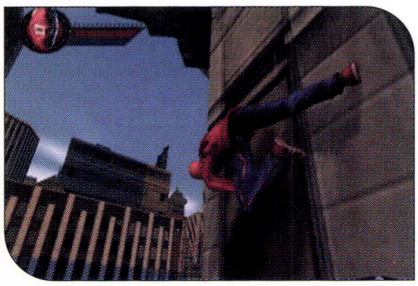
brew™
Customize. Personalize. Realize.™

Put your imagination in motion. The revolutionary wireless platform BREW enables developers – even those with no wireless experience – to write applications and get them to market. No finding a handset partner. No gaining access to chip system software. BREW does it all. Download the BREW SDK at http://brew.qualcomm.com/ZGD3. Wireless carriers need apps. Start thinking.

©2002 QUALCOMM Incorporated. All rights reserved. QUALCOMM is a registered trademark of QUALCOMM Incorporated. BREW and Customize. Personalize. Realize. are trademarks of QUALCOMM Incorporated.

# INDUSTRY WATCH

THE BUZZ ABOUT THE GAME BIZ | daniel huebner



The success of SPIDER-MAN: THE MOVIE games may help Activision reach new revenue targets.

**Xbox cuts price, lowers estimates, and loses staff.** Six weeks after Xbox made its European debut to uniformly sluggish sales, and amid vocal criticism from consumers and publishers alike, Microsoft chose to reduce the console's pricing. The company cut the cost of an Xbox on the continent from 479 euros to 299 euros, putting Xbox's price tag square with that of Playstation 2.

Six weeks of slow European sales at the higher price, combined with a similarly lackluster Japanese launch, had already done enough damage to force Microsoft to revise its Xbox sales estimates. Microsoft had expected to ship 4.5 to 6 million Xbox consoles by the end of June 2002 but now puts that number closer to 3.5 to 4 million.

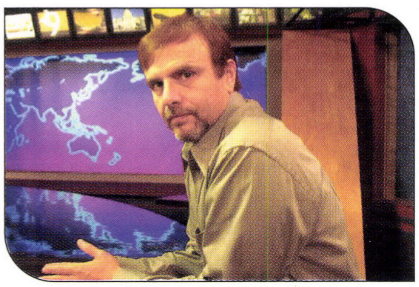
New pricing and lowered estimates were followed by an even greater change to Microsoft's Xbox efforts: the departure of Seamus Blackley. Blackley, credited as one of the creators of the Xbox, insisted that the timing of his departure was coincidental and motivated by the desire to return to a position more closely related to making games. His next role will be that of VP of development for the newly formed Capital Entertainment Group. CEG, launched by fellow Xbox alum Kevin Bachus, hopes to carve a niche for itself by helping developers get the financial and production resources necessary to bring their games to market.

**Interplay sells Shiny to stay afloat.** Interplay has dodged bankruptcy once again, but the company lost one of its most promising assets in the effort. Herve Caen, Interplay's CEO, announced in mid-April that the company was on the verge of bankruptcy and seeking to raise operating funds by selling off its Shiny Entertainment unit. Caen further warned that money raised by the Shiny sale would only serve as a temporary solution and that the company's operating losses, deficits in stockholders' equity, and lack of working capital raised substantial doubt about Interplay's ability to continue as a going concern.

Infogrames agreed to purchase Shiny for approximately $47 million in a combination of cash and a promissory note. That price represents a substantial premium, as Interplay acquired an initial 91 percent of Shiny in 1995 for $3.6 million in cash and stock, and then picked up the remaining 9 percent in March 2001 for $600,000. As a result, Infogrames will have exclusive worldwide rights to develop and publish games based on sequels to the movie *The Matrix*, the first of which is currently in development by Shiny. Infogrames also gets Shiny's patent for "Advanced Tessellation Technology." Dave Perry, founder and president of Shiny, signed a long-term employment agreement with Infogrames to remain as president of Shiny.

**EA, Activision report surging fourth-quarter revenues.** Electronic Arts crushed analysts' forecasts for the fourth quarter by posting an impressive 53 percent increase in quarterly revenue. Revenues expanded to $469.7 million from $307.3 million a year earlier, and the surging sales pushed profits to $47.3 million — reversing a $17.9 million loss in the same period last year. The company's faltering online division, blemished earlier in the year by high-profile layoffs and the cancellation of MAJESTIC, saw improved numbers but was unable to reach profitability. EA.com nearly doubled its revenue, to $23.6 million from $12.8 million last year, but still posted a fourth-quarter loss of $32.9 million.

Activision experienced a similar jump in sales, as revenues in its fourth quarter grew by 30 percent to $164.9 million from $126.8 million last year. Net profit for the quarter totaled $10.9 million, a sizeable gain from last year's $875,000. Xbox was Activision strongest console platform for the quarter, accounting for 21 percent of publishing revenue. In light of the results, Activision has raised its guidance for 2003 and plans to target $1 billion in revenue in 2004 — joining EA as the only publisher to realize $1 billion in annual revenue.



Profitability continued to evade EA's online division in the fourth quarter, despite efforts such as the cancellation of MAJESTIC.

## UPCOMING EVENTS CALENDAR

### LINUX WORLD EXPO

MOSCONE CONVENTION CENTER
San Francisco, Calif.
August 12–15, 2002
Cost: $10–$1,050 (early bird discounts available)
www.linuxworldexpo.com

### GDC EUROPE

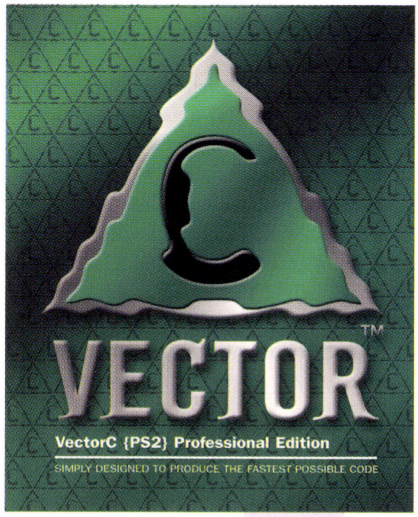
EARL'S COURT
London, U.K.
August 27–29, 2002
Cost: £350–£450
www.gdc-europe.com

### ECTS

EARL'S COURT
London, U.K.
August 29–31, 2002
Cost: Advance registration free via web site; £25 on-site registration
www.ects.com

{ADVERTISEMENT}

# Unlocking the power of the PlayStation®2




## Secrets of the PlayStation®2

Computer games are all about floating-point performance. With 89% of the floating-point power of the PlayStation®2 locked up in the vector units, these co-processors are often an under-utilised resource.

VU1, with its fast pipe to the GS, is typically used for simple final-stage transformation and lighting pipeline programs.

Codeplay's VectorC {VU} allows programming the PlayStation®2 vector units in C/C++ and this allows smarter, more complex rendering pipelines to be implemented which cut down on DMA traffic and allow for new and improved visual effects.

However it is VU0 which is the best kept secret to obtaining extra performance on the PlayStation®2.

Many PlayStation®2 developers utilise the 'VU0' vector unit to accelerate floating-point math in code such as physics, AI and collision detection. Typically by embedding in-line macro-mode assembly within their C++ code compiled for the EE core.

However a significantly higher performance approach is to write micro-mode vector unit programs running asynchronously on VU0. The benefits of this approach are;

- Both EE core pipelines are able to do work at the same time.
- VU0 is used in its native dual-issue mode with fast local memory.
- The EE core instruction cache is left free for EE core instructions.
- No need for in-line assembly, avoiding further compiler performance issues.

## So why has it not been done?

The reason that micro-mode programming of VU0 is rarely undertaken is because of the difficulty in writing VU assembly language and handling the asynchronous timing of a parallel function running on VU0 at the same time as the EE code.

## Codeplay's VectorC {VU} solves both these problems

VectorC {VU} allows you to move your original code off the EE completely without re-writing in assembly language. It does this by allowing rapid programming of the vector units in C/C++.

Secondly, VectorC {VU} provides header files that allow calling of your micro-mode functions as if they were local functions. The process of passing data to the function and retrieving results later is done transparently by macro-mode.

## How does this work?

Any function in your game which requires a relatively small amount of data but which is heavy on floating point calculations is suitable for moving off the EE and onto VU0. This code is compiled with VectorC {VU} which in turn generates a header file which is included in the main EE code. From now on the function now implemented on VU0 can be called as if it is a local function. Data will be transparently passed to the VU0 micro-mode program and the program executed.

While the VU0 program does its work, the EE is now also available to work such as loading the next set of data. After that, another function may be called to fetch the results from the VU0 micro-mode program.

```
---- vu0demo.cpp:
float A (float x)
{
    return x / 3.0f
}
```

```
---- eecore.cpp:
#include "vu0demo.vu-h"

float B (float x)
{
    return x / 4.0f;
}

void Test ()
{
    float s, t;
    A (5.0f);
    t = B (6.0f);
    GET_RESULT_A (s);
}
```

In the simple example shown, the functions A and B are executed at the same time. A is executed on VU0 and B is executed on the EE core.

The result is relieving the EE of work better suited for the vector units which translates to a sizeable improvement in performance. All without changing the fundamental way that your game code is written.

More features or a faster game, you decide!

**VectorC {VU} is available now.**
**www.codeplay.com**

# A Tale of Two Workstations
# 3DBoxx and Dell Precision Mobile M50

by david stripinis

The term workstation is often used with some alacrity to describe the computers game artists work on every day. Few, however, would truly be considered an heir to the title passed down from the SGIs of yesteryear. This month I took a look at two products that both live up to the title and at least one that redefines the way CG artists can work.

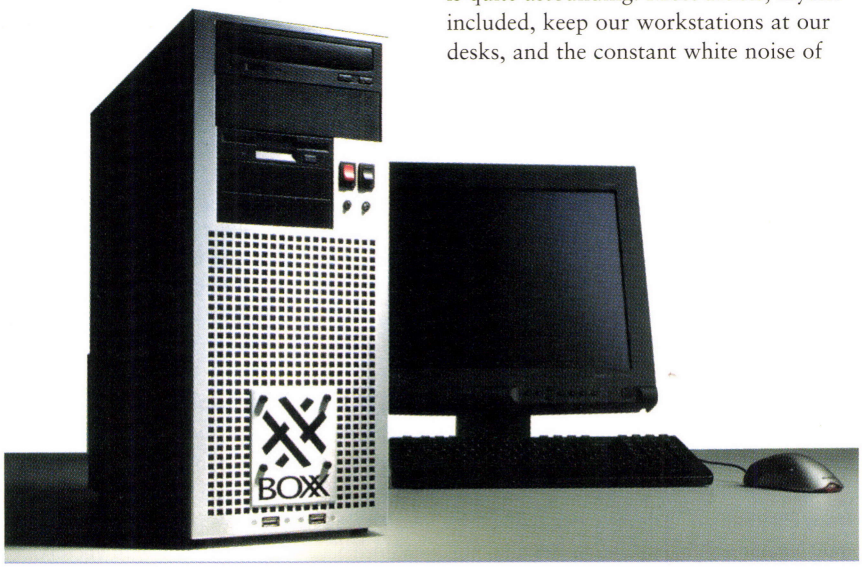
## Boxx Technologies' 3DBoxx

First up is Boxx Technologies' 3DBoxx, featuring dual AMD Athlon 2000+ processors, 1GB 266MHz DDR memory, an Nvidia Quadro 4 900 XGL 128MB DDR video card (which supports two monitors), and an 80GB hard drive. Though it comes equipped with two Ethernet ports, there is no Firewire port as a standard feature, which is unfortunate, as any modern graphics workstation should be able to support DV camcorders, a variety of hard drives, and other peripherals that need the superior bandwidth of Firewire over USB.

It's hard to look at a desktop workstation these days and see its true value. After all, in the world of www.cheappartsforyourpc.com and with the geekish nature of game developers, we feel we could make more for less than buying a turnkey solution. And if you perceive your purchase as merely the buying of parts, you may be right. But what is the value of your time and sanity? Would you rather spend weeks researching which motherboard works with which processor (and with how much RAM) and then trying to find a good video card that doesn't have issues with all of the above and, on top of it all, performs well with your software? Or would you rather just take a machine out of the box, plug it in, and go to work? With a machine like the 3DBoxx, that's precisely what you do. Throw in one year of 24-hour guaranteed part replacement, and suddenly that extra premium you paid is looking better.

Upon unpacking the 3DBoxx, two things struck me. First was its weight (or lack thereof). While it feels quite solid, the case and mountings are made of aluminum, rather than traditional steel. Not only does this make the machine more portable, but, given the qualities of aluminum, it turns the entire case into a gigantic heat sink, dispersing the tremendous amount of heat given off by today's high-end processors. The second and definitely more noticeable aspect of the machine is the industrial design of the case itself. With its retro black-and-brushed-metal look, dipswitches, and gigantic, bright LEDs, the machine just looks powerful. But it also manages to stay traditional enough so you don't feel you are paying a premium simply for the case design.

After I plugged everything in, I hit the power button. In a seeming assault on my senses, the bright power LED and cooling fans came to life. Now, I realize the need to keep the chips inside cool, and the simple physics of fan blades chopping through the air at high speed are bound to cause some noise, but the amount of racket this machine puts out is quite astounding. Most artists, myself included, keep our workstations at our desks, and the constant white noise of




**DAVID STRIPINIS** | *David is currently on the third year of his life sentence as Factor 5's animation monkey. Care packages and questions can be sent to david.stripinis@factor5.com.*

the fans began to grate on me after a while. It would be interesting to see a commercial workstation vendor like Boxx start using one of the existing liquid cooling systems.

Perhaps the noise level is justified, however, because this machine just screams. Twenty minutes out of the box, I had both Maya and Lightwave installed and running. This is one of the main arguments for purchasing a workstation, rather than a bunch of parts: Everything not only worked (and worked well), but also worked well together. I had no device conflicts, driver issues, or odd dipswitches to set, and this on a Windows 2000 machine (which is the preinstalled OS, though I had no trouble installing Red Hat Linux as a dual boot solution).

The Nvidia Quadro 4 video card is a true beast, pumping out a 3DMark of 8820 at the default settings. Complex scenes in Maya with GL fog, shadows, and large amounts of textures were swallowed whole. Because the Quadro 4 supports hardware overlay planes, many of Maya's features that rely on them, such as 3D Paint, really displayed the prowess of the system in a production environment. Unless you're working on some super game platform unknown to man, this system should handle any modern videogame art asset without a problem. The only issue I had within both Maya and Lightwave was a tendency for viewports in the background to not be updated. This is hopefully something that can be rectified within the next few driver releases from Nvidia.

## Dell Precision Mobile Workstation M50

The other machine I had a look at was the Dell Precision Mobile Workstation M50. For such a corporate-sounding name, this is one sexy machine, and without a doubt one of the most impressive pieces of hardware I have had the pleasure of using. My test system held a Pentium 4 running at 1.8GHz, 512MB of DDR memory, a 40GB hard drive, and an Nvidia Quadro 4 500 Go GL 3D with 64MB DDR memory. This is a true portable workstation. I hesitate to use the word laptop, because the heft of this system, as well as the heat it generates, would give pause to anyone wanting to keep this machine on his or her lap.

Featuring a crisp, clear 15-inch display that runs at a native 1600×1200 resolution, the machine is equipped with a bevy of ports. In a taunting game of one-upmanship with the 3DBoxx, the Dell does feature a Firewire port. Lacking, however, was an integrated 802.11b networking solution. Any portable in this class should feature such technology as standard, rather than as the option it is available.

My other major gripe was the pointing devices integrated into the keyboard. Featuring both a touchpad and a pointing stick, the users have a choice as to which way they wish to navigate around the UI. However, neither features a scroll wheel or third button. Most of the applications I run in my day-to-day existence require that middle mouse button, so I found myself carrying around an external mouse.

O.K., enough with the griping and on with the good stuff. Putting it simply, this machine is unbelievable. Sure, laptops have served as desktop replacements for years now, but the thought of a portable that would run a 3D program, never mind run it faster than 99 percent of the computers people work on every day, is astounding to me. Running Maya on it was a dream, and earned oohs and aahs from my coworkers. I could easily tumble around complex models numbering in the hundreds of thousands of polygons. I could run cloth simulations in near real time and render out animations with surprising speed. Hard drive access was a little slow, but a slower hard drive lets the M50 conserve power, giving it a respectable average battery life of two hours and 12 minutes of actual use. Dell includes a handy utility that gives a fairly accurate display of remaining battery life in time, rather than a percentage as many similar utilities do.

I didn't need to take any long trips while I was conducting the review, but I can definitely see the value of having a machine of this caliber as a portable. The ability to run all your tools and programs wherever you go, be it a recruiting trip, a professional conference, or just going outside and working in the fresh air for a while is truly worth the premium you pay for portability.

Also equipped with an Nvidia chipset, the M50 performed well against the Boxx in the 3DMark, scoring an impressive 4855. Interestingly, neither 3DMark nor Right Hemisphere's Deep Exploration recognized the video chip as anything above a GeForce 2, so I could not use any of the pixel shader features of Deep Exploration nor run the pixel shader tests of 3DMark. Again, I hope this can soon be rectified through a driver release.

## Max Power's the Name

If you are in the market for new workstations for your staff, I can't recommend the 3DBoxx highly enough. Though the noise can get a little distracting, the intense, raw power it brings to the table makes it worth it. Maybe we need to bring back the days when workstations were kept in a super air-conditioned room well away from the animator? You pay a premium for service, but

★★★★★ excellent
★★★★ very good
★★★ average
★★ disappointing
★ don't bother

with 24-hour part replacement and the personalized service Boxx offers, it's more than worth it, especially if your staffing budget doesn't allow for full-time IT support.

If you need to hit the road often, or you just like the idea of truly being able to take your work home with you, the Dell M50 is worth every penny and every pound. Sure it's heavy, lacks the essential third mouse button, and will scorch the hair off your legs if you actually put it on your lap, but it's really a true workstation disguised as a laptop. While I'm waiting for Dell to ask for its return, I am in fact writing this review on it. They'll get it when they pry it from my cold, dead hands.

Looking at these two machines, both of which retail for less than what I paid for a 286/16MHz 10 years ago, all I can say is Moore's law is really, really cool.

## 3DBOXX ★★★★

**STATS**

BOXX TECHNOLOGIES
(877) 877-BOXX (2699)
(512) 835-0400
www.boxxtech.com
PRICE (AS CONFIGURED)
$3,527 (MSRP)

**PROS**

1. Powerful.
2. Affordable.
3. Great service plan.

**CONS**

1. Noisy.
2. Case design is a love it/hate it situation.
3. Lacking in I/O abilities.

## DELL M50 ★★★★

**STATS**

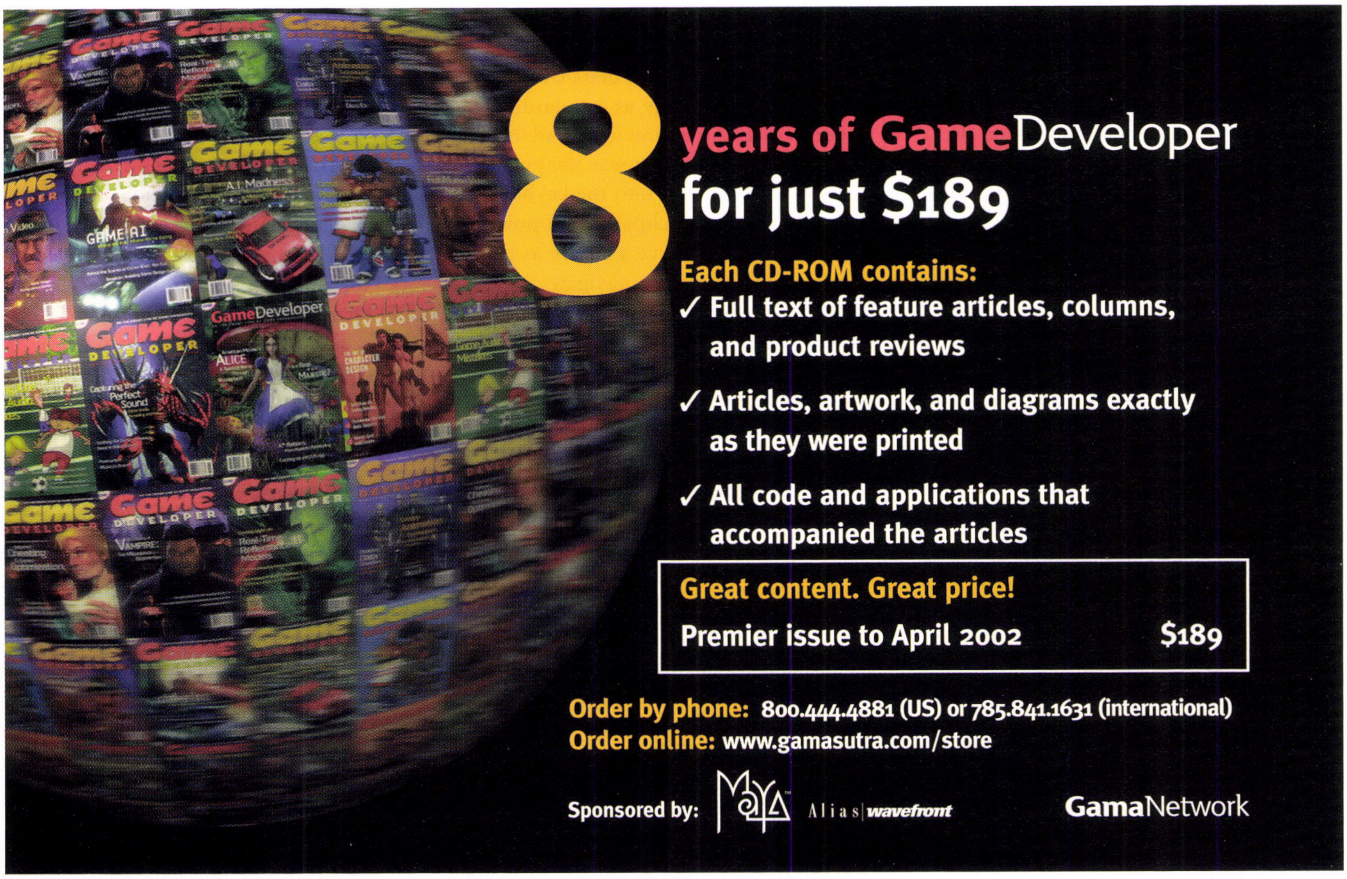
DELL COMPUTERS
www.dell.com
PRICE (AS CONFIGURED)
$4,343 (MSRP)

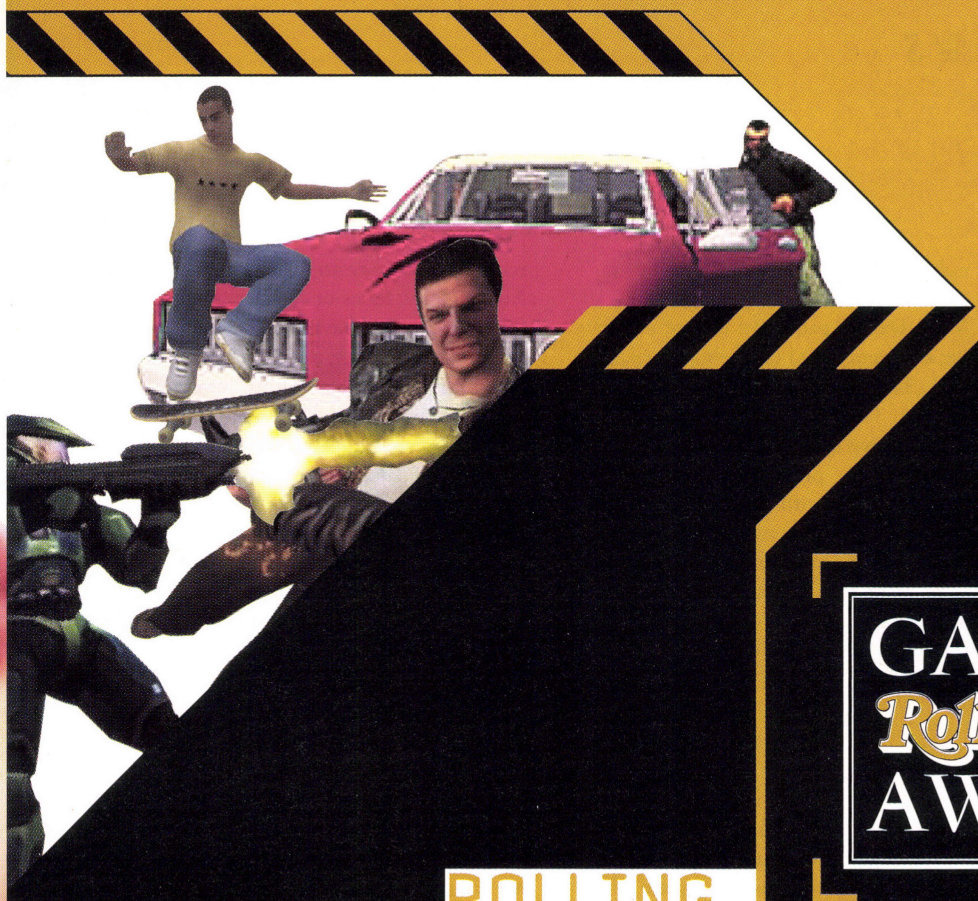
**PROS**

1. Powerful enough to be a desktop workstation replacement.
2. Portable.
3. Beautiful high-resolution screen.

**CONS**

1. Lack of a third mouse button on integrated devices.
2. No 802.11b as a standard configuration.
3. More expensive than what you'd pay for a comparable desktop of the same capabilities.





ROLLING STONE CONGRATULATES THE WINNERS OF THE 1ST ANNUAL ROLLING STONE GAMING AWARDS

# GAMING Rolling Stone AWARDS

## CONSOLE GAME OF THE YEAR
GRAND THEFT AUTO 3
ROCK STAR GAMES

## PC GAME OF THE YEAR
MAX PAYNE
ROCK STAR GAMES/
GATHERING OF DEVELOPERS

## BEST MUSIC
ORIGINAL SCORE
HALO
MICROSOFT

## BEST MUSIC
LICENSED SOUNDTRACK
TONY HAWK'S PRO SKATER 3
ACTIVISION

Rolling Stone reaches 4.4 million owners of gaming consoles and 7.2 million households with computers every 2 weeks*

For more information about the Awards, and a copy of The Rolling Stone Gaming Issue (#897 cover date 6/6/02) contact: Rolling Stone Magazine, Associate Publisher, Michael Hess, at 212-484-3415

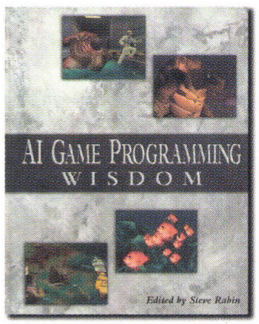
*Source: Fall 2001 MRI Base: Adults

★★★★★ excellent
★★★★ very good
★★★ average
★★ disappointing
★ don't bother

## AI GAME PROGRAMMING WISDOM

**Edited by Steve Rabin**

*reviewed by Soren Johnson*




Following in the footsteps of the now-venerable *Game Programming Gems* series, *AI Game Programming Wisdom* attempts to explore what is perhaps the least understood of all game development disciplines, artificial intelligence. Like the *Gems* series, this book is composed of short chapters (ranging between five and 10 pages each) focusing on solutions to common AI challenges. Typical entries include "Realistic Turning Between Waypoints," "Camera AI for Replays," "The Beauty of Response Curves," and the ever-popular "Intercepting a Ball." Indeed, the sheer scope of the book is impressive; entire sections are devoted to A* pathfinding, group movement, scripting, and even sports-specific AI.

The gems themselves are a mixed bag. Some detail powerful techniques for increasing the realism, adaptability, and fun factor of game AI, while others, such as "Architecting a Game AI," are simply too high-level to be valuable, regardless of the author's honesty and good intentions.

The general problem lies within the old debate of whether AI is an art or a science. The truth, of course, is that good game AI requires both creative vision and solid engineering. Thus, the best articles are grounded in real-world experience resulting from this synergy. For example, Stainless Steel Studios' Dan Higgins contributes an excellent chapter on building a generic "A* machine" for EMPIRE EARTH, which, besides controlling pure pathfinding, also handled weather models, wall building, terrain analysis, and choke-point detection. Even the relatively nonspecific "12 Tips from the Trenches," by Monolith's Jeff Orkin, is useful for illuminating common AI stumbling blocks.

Bioware's Mark Brockington and Mark Darrah provide a pair of interesting chapters based on their experiences with BALDUR'S GATE and NEVERWINTER NIGHTS. The first one describes a simple but effective level-of-detail system for providing each NPC AI with appropriate processor time depending on its proximity to the users. The second details "How Not to Implement a Basic Scripting Language," emphasizing the need for designing a syntax appropriate for the script authors and planning for extensibility from the beginning.

The book also includes excellent building blocks for beginning AI game programmers. The companion CD includes extendible source code for an A* pathfinding algorithm, a finite state machine, a rules-based inference engine, and a fuzzy logic library. The accompanying chapters do a good job of preparing the developer to dive into the code. The A* project even includes a fun little program for viewing the algorithm in action on a user-modifiable map.

Genetic algorithms and neural networks, the redheaded stepchildren of the game AI community, are confined to a few blandly general chapters at the back of the book. Perhaps someday these techniques will become mainstream AI techniques. The path to that day, however, is not laid out in this book.

The book does suffer from a lack of articles on AI-based gameplay. Lionhead's Richard Evans supplies an informative, if too short, article on the AI architecture behind the creatures in BLACK & WHITE. Perhaps the next version could dedicate an entire section to this topic, including articles on CREATURES, THE SIMS, and THIEF: THE DARK PROJECT.

Another area to improve is diversity. Just seven authors are responsible for 26 chapters, representing over one-third of the total content. Of the five chapters covering A*, for example, three are written by the aforementioned Dan Higgins. Although his series of chapters are uniformly excellent, the reader would be better served by having a greater variety of perspectives on the strengths, weaknesses, and subtleties of the A* algorithm.

So, who should buy *AI Game Programming Wisdom*? For inexperienced game developers — which describes a bigger portion of our industry than most would care to admit — the book is absolutely essential reading. Nowhere else can so much basic wisdom be found in such concentrated form. The pathfinding sections alone make the book a simply invaluable purchase for beginners. Veteran AI programmers, however, might be disappointed. Unless the reader works within one of the AI subdisciplines in which the book really excels, such as racing AI or tactical group movement, the collection of articles is unlikely to fundamentally change how he or she works.

Still, as a collection of this scope is a first for our young industry, it would surely be a worthwhile purchase for almost anyone involved with game AI. The discipline certainly suffers from a lack of standards. Like the cursed Sisyphus spending eternity rolling his rock up a hill only to see it fall down the other side, game AI programmers seem perpetually doomed to throw out their best work and start from scratch with each new project. Until this cycle is broken, high-concept AI work, such as learning and adaptation, will be confined to academia and games ambitiously conceived by Peter Molyneux. *AI Game Programming Wisdom* represents another small but significant step toward an established game AI community, with shared algorithms, building blocks, and architectures. Perhaps the industry is finally growing up.

★★★½ | *AI Game Programming Wisdom* | Charles River Media | www.charlesriver.com

*Soren Johnson was the co-designer and lead programmer of Civilization III. He can be reached at sjohnson@firaxis.com.*

Photo by Bill Wiegand, University of Illinois

# Put Your Game Online
# We'll Keep It There

**Windows® * PlayStation® 2 * Nintendo GameCube™ * Xbox™ * Macintosh® * Linux®**

gamespy.net

We'll take care of everything: technology, servers, support, even bandwidth costs.

Getting rock-solid player matchmaking, chat, and server browsing in your game is only the first step. When you work with GameSpy, you get more than solid, easy-to-implement technology. You also get reliable multiplayer back-end support, 24/7, for the lifespan of your product.

We've got over six years of multiplatform online gaming experience. You make the game, we'll keep it running.

**Find out more:**
**www.gamespy.net**
**bizdev@gamespy.com**
**Art Santos - (949) 798-4267**

POWERED BY gamespy™

Technology to put your game online.
Service to keep it there.

Online solutions for Windows®, PlayStation® 2, Nintendo GameCube™, Xbox™, Macintosh® and Linux®

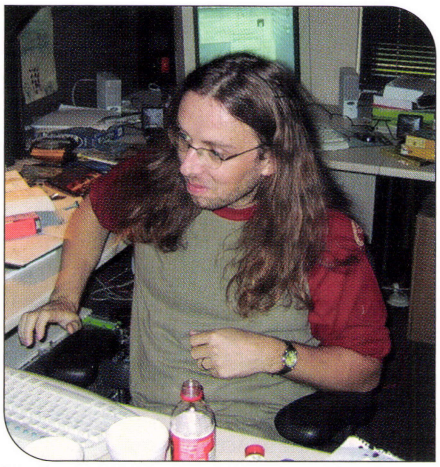
- Peer-to-Peer Player Matching
- Server Browsing
- Instant Messaging and Buddy Lists
- Text and Voice Chat
- Online Storage of Game Data
- Competition and High-Score Systems
- Online Anti-Piracy
- Usage Analysis and Reporting
- Cross-Platform Networking
- Auto-Patching
- Online Data Transfer

Copyright © 2002 GameSpy Industries, Inc. GameSpy, Powered By GameSpy and the SpyGuy Logo are trademarks of GameSpy Industries, Inc. , All rights reserved.
All other trademarks and names are properties of their respective owners.

# Devine Revelations

## A conversation with id's Graeme Devine

Once upon a time, Graeme Devine was an ordinary British high school kid cutting class to do ports of games like POLE POSITION for Atari. He immigrated to the U.S. and later co-founded Trilobyte in 1990, which released the seminal 7TH GUEST and its ambitious sequel, THE 11TH HOUR. These days he is best known for the many hats he wears at id Software, where he has worked as a game designer, programmer, project manager, and self-described "Mac guy," among other things. *Game Developer* recently interrupted his work on DOOM 3 to get some perspective from a true industry veteran.



id's Graeme Devine hard at work on DOOM 3.

***Game Developer.*** **You've spent a year on the IGDA board of directors and recently became chairman of the board. What do you feel like you've accomplished, and what do you hope to accomplish still?**

**Graeme Devine.** I think it's actually done more for me than I the other way around. I've come to respect greatly the community, thought, and care that the IGDA drives through its members. For me, the big thing in the last year has been that we've plastered schools with information on how to get into the game industry and what sort of skills we look for. For years after I worked in this industry my friends and family all thought of it as a fad job — something you grow out of. They would often nudge me to go get a real job. Hopefully the IGDA push here will change that perception, and therapists around the world will hear less from stigmatized game developers.

***Game Developer.*** **You've been working on games for almost a quarter of a century. It's easy to tell what's changed about game development in that time, but is there anything that hasn't changed?**

**Graeme Devine.** I think we all get caught up in technology so much now that we forget that the actual game stuff, the fun part, the thing you kick, hasn't really changed much at all. Or perhaps it was actually more broad way back compared to nowadays.

***Game Developer.*** **If we assume that technology is becoming less of a distinguishing factor among different games out there, how is that changing the relationship between future technology and future gameplay opportunities?**

**Graeme Devine.** I think the main deal here is that people will actually have to get back to making something compelling as well as something that looks darn cool on that new piece of hardware you've just purchased. Well, that's what I'd like to happen. What I actually think will happen is game content will become harder and harder to produce because photorealism is hard to make. Studios will change their models. For example, small studios will prosper as providers to a game that other studios are also working on. Cats and dogs will live together and we'll still end up alone on Friday nights.

***Game Developer.*** **How much programming do you get to do these days?**

**Graeme Devine.** I actually do quite a bit. I just finished this really cool thing that does some really cool stuff for DOOM. Of course I can't tell you about it, so you'll have to believe me at my word. Actually, most of what I do at id these days is program. Although, of course, you can't stop a game designer designing.

***Game Developer.*** **Do you subscribe to any game design philosophies? Any method to the madness?**

**Graeme Devine.** Are we old enough yet to have design philosophies? I suppose I try and think to myself about how to get past an objective without shooting it. Which, while explaining my poor QUAKE 3 skills, is actually a very good question to apply anywhere in a game where you play and progress via killing. There are so many more interesting ways.

Most of the games I design are world first, game second. I love making worlds and making them work. What is the language, how do they work, why do they work, and make that logical and believable. Then I'll write a few short stories set in the world to see if it makes sense to me, and if a story can be interesting, then the game comes naturally and feels solid. Of course, that's just me.

***Game Developer.*** **What kinds of game design inspiration do you find out there in the real world?**

**Graeme Devine.** Comics and films. And sometimes, just sitting in Starbucks watching the world, you see the strangest glimpses of other lives that your mind expands out to make some kind of bizarre, twisted world.

***Game Developer.*** **Besides John Carmack, who's the smartest person at id?**

**Graeme Devine.** That would be John's clone. We had him cloned last year so we could get double the amount of work he does. So far, it's worked out pretty well except for the huge amount of diet cola the two of them consume.

# What are you waiting for?

Want to work on the most sophisticated piece of 3D graphics hardware in existence? Well get in line. Everyone wants a Wildcat® III. Features alone make it the most coveted piece of hardware in any design environment. You get a truckload of memory. Ultra-realistic image quality. Plus, application performance feels like somebody's plugged a supercharger into your workstation. Lucky for you, Wildcat III 6210 and 6110s are available in just about every professional workstation. Getting yours couldn't be easier. So what are you waiting for? Log on to **www.3dlabs.com/whywait** for more information.

| | |
|---|---|
| 416/208 MB DDR RAM (6210/6110) | Dual Monitor Support |
| SuperScene™Full Scene Antialiasing | Six T&L Engines with 32 Light Sources |
| Supports OpenGL® 1.3, OpenML™ 1.0 & DirectX® 7.0 | Multiview & Genlock Support (6210 only) |

Wildcat III

www.3dlabs.com

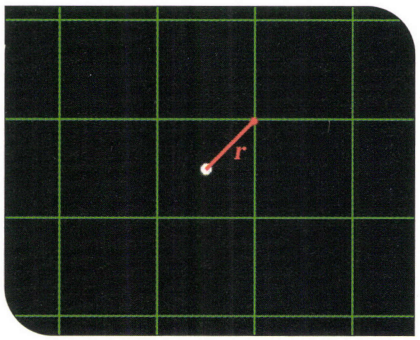
3Dlabs, Wildcat and SuperScene are trademarks or registered trademarks of 3Dlabs Ltd., 3Dlabs Inc. Ltd., or 3Dlabs Inc. in the United States and/or other countries. OpenGL is a registered trademark and OpenML is a trademark of Silicon Graphics, Inc. DirectX is a registered trademark of Microsoft. All other trademarks are acknowledged and recognized.

# Transmitting Vectors

This month's column continues our ongoing discussion of packing values into small spaces, which is useful for network communications and save-games. Previously, I discussed integers ("Packing Integers," May 2002) and scalars ("Scalar Quantization," June 2002); this month I'll talk about vectors.

As game developers, we most commonly deal with two different categories of vectors. The first category is unconstrained vectors, with "unconstrained" meaning that none of the vector's coordinates can be written as an equation of its other coordinates. An example is the position vector of an entity in the game world; we may set limits on the magnitudes of the X, Y, or Z components, but otherwise, they are independent.

The second category is unit vectors, which are spherically constrained. Examples are surface normals, direction vectors (perhaps representing which way a player is looking), and rotations (which are unit vectors in four dimensions).



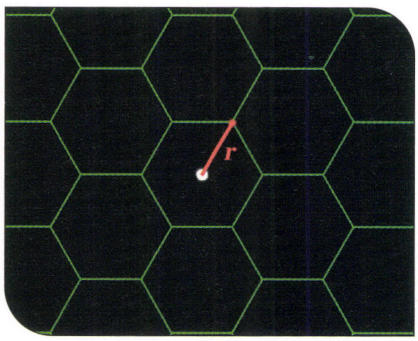




FIGURE 1 (left). Tiling the plane by squares of area 1. The point on a square farthest from its center is about .707 units away. FIGURE 2 (right). Tiling the plane by hexagons of area 1. The point on a hexagon farthest from its center is about .620 units away.

## Unconstrained Vectors

I'll start by looking at a 2D position vector. The most obvious (and most common) way to quantize the vector is to treat the coordinates as two separate scalars and encode them using a method from last month, probably the vanilla scalar quantization. This approach is equivalent to dividing the map up into a bunch of grid squares, finding which square contains your entity, and using the center of that square to represent the entity's position.

The resolution of that grid controls the quality of our encoding; if the grid squares are too big, we won't be able to represent positions accurately. Usually we choose a grid resolution by determining how much error we can tolerate and then finding the square size that gives us that much error at maximum. The most an entity's position can be wrong via this encoding is the distance from the center of the square to one of its corners (Figure 1).

This technique has some disadvantages. One problem is that the accuracy of transmitted entity positions is highly anisotropic. If an entity is moving along a diagonal of the grid, its average error will be much higher than an entity moving along an axis. Anisotropy is nice in rendering, but here it is a bad thing. It's bad because if we make the grid squares small enough that the error in diagonal directions is acceptable, then we get more accuracy than we need in the axis-aligned directions. That means we've wasted some bandwidth.

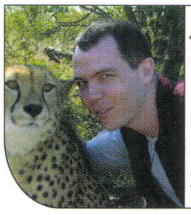
We can reduce this effect by quantizing into shapes that are rounder than squares. War-gamers will be familiar with the tiling of the plane by hexagons (Figure 2). A square of area 1 has extremal points about .707 units away from the center; a hexagon of area 1 has extremal points about .620 units from the center. That's a little more than 10 percent closer.

This implies, via a little bit of math, that you need 30 percent more squares than hexagons to tile an area with equal error thresholds. So when using hexagons, you save about .4 bits per transmitted position. That might sound small, but if you have a bandwidth bill the size of a successful MMORPG's, you've saved perhaps $2,500 a month.

(Statistically-minded readers can compute the covariance matrices for the square and hexagon and observe the magnitudes of the eigenvalues, which tells you their relative compactness.)

We might try to do better than hexagons by seeding the plane with a bunch of unstructured points and assigning all input values to the closest point. This approach is equivalent to tessellating the plane by the Voronoi diagram of those points and choosing the Voronoi region in which an entity lies.

We want the points to be as evenly spaced as possible. We could precompute them by running an energy relaxation program that causes points to repel each



**JONATHAN BLOW** | *Jonathan (jon@bolt-action.com) is a game technology consultant living in San Francisco. This month's article was turned in late to his editor because Jonathan was playing Piranha Bytes Software's GOTHIC, one of the most interesting and fun RPGs made in a long time. Jonathan encourages you to try the game out if you have any impending deadlines.*

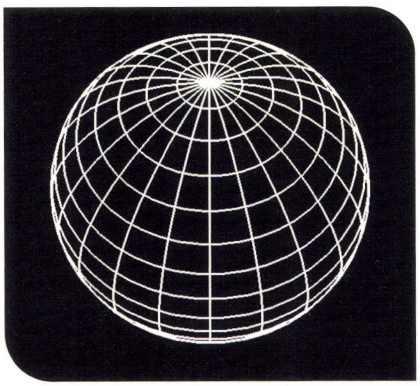


FIGURE 3. When we quantize a sphere parameterized by azimuth and elevation, we get the patches shown here. The patches near the poles are radically different in size from those near the equator; this is bad.

other until they reach equilibrium. I have included such a program in this month's sample code (available for download from the *Game Developer* web site at www.gdmag.com). When you run the program, you get . . . hexagons.

It turns out that the problem of tessellating the plane into congruent shapes is equivalent to the circle-packing problem. (At a coarse level, you can think of the relationship this way: We are trying to tessellate the 2D plane with compact shapes, and a circle is the most compact 2D shape. We want to minimize the amount of the plane that ends up between the circles, as these portions produce higher-than-ideal error.)

A lot of smart people have already spent a lot of time thinking about circle packing, which is good for us. It's been proven that, in 2D, the most optimal way to pack circles is in a hexagonal lattice (see For More Information). Translated to the task at hand, this means that there is no better way to quantize the 2D plane evenly than to use hexagonal tessellation (when I say "better" I am judging with respect to the error metric I outlined earlier).

At this point we may wish to think about why crystals like to form in hexagonal shapes, or why bees build honeycombs as a hexagonal tiling. These kinds of phenomena help maintain my faith that game programming is teaching me deep and important things.

## Expanding to 3D

Suppose we want to quantize 3D space. We can turn again to the circle-packing guys for an answer to this question. As it happens, there are two equally good packing shapes for 3D, both of which are proven to be optimal among regular lattice packings and are very strongly believed to be optimal as general packings as well. One of these shapes is a 3D extension of the 2D hexagonal tiling. The other one, which I find easier to visualize, is called the face-centered cubic packing. You can find out lots of relevant things about circle packing just by typing "face-centered cubic packing" into your favorite search engine.

However, direct 3D quantization is probably overkill for most games. For example, most MMORPG worlds are predominantly two-dimensional, and most things in the world are resting on a solid surface. So a more efficient way of transmitting position is to use the 2D hexagonal tiling to communicate the entity's XY position and then use a vertical raycast to compute a small integer index for which surface the entity is standing on.

When you use this method, however, you have to provide some handling for exceptional cases, such as when the triangle on which the entity is standing doesn't extend all the way to the XY of the quantized position.

You can Huffman-code these vertical triangle indices, so when something is supported by a very popular triangle, the Z information requires only 1 bit to transmit. When an entity is flying or swimming, you can use a fallback method of transmitting position that consumes more bandwidth in order to explicitly communicate Z.

## Constraints

So you see that before we even introduce constraints, the two dimensions are tied together somehow — we can't really treat them independently without suffering a bandwidth hit. The dimensions are intertwined by a requirement of our problem domain, namely that we want some kind of relativity with respect to orientation. That is, we want isotropic behavior so that the reference frame can rotate and our world will not behave much differently. At some level I think this is a deep concept, though when stated like this, it's a "well, duh" kind of thing. Mathematically, this comes down to the fact that we're using the Euclidean norm as our distance metric, and the Euclidean norm ties dimensions together.

Next I'll discuss constraints, which tie things together even more. First, though, I'll clarify that the general process we are performing here, mapping an input vector to some vector in a fixed dictionary, is known as vector quantization, or VQ. VQ has been studied extensively, and there's a lot of literature out there about it. I'm not going to philosophize about general VQ here; I'm just looking at a couple of very specific cases.

## Unit Vectors

The most popular kind of constrained vector in games is the unit vector, either in 3D (to represent things such as aiming directions or surface normals) or in 4D (to represent rotations). I'll start with 3D.

The most common ad hoc method of quantizing 3D unit vectors is to convert them into azimuth-elevation form and uniformly quantize each parameter separately. This technique is awful. When you use it, you get Figure 3. In order for the patches near the equator to be acceptably small, there needs to be a big excess of patches near the poles. We're wasting bandwidth.

The set of all 3D unit vectors is equivalent to the set of points on the unit sphere in 3D. So to perform VQ, we want a dictionary consisting of points evenly spaced on the unit sphere.

In 2D, it was easier to visualize the dilemma by solving the equivalent problem of tessellating the plane. But on the unit sphere in 3D, the situation is less intuitive. We could start solving some math about how to create a maximally uniform tessellation of the curved surface of the sphere (or hitting the math books, as I'm sure several people have worked this one out before), but that would be a

big pain. We could also settle for some shape that seems reasonable, such as a soccer ball. However, this solution doesn't give us any control over how many dictionary vectors we get to use (the resolution of our encoding), and we also don't know how close to optimal it is for a given dictionary size.

The simplest solution for us is just to run another relaxation program, using the dot products between vectors as a metric that tells us how much they repel each other. I've included such a program in this month's sample code. You use a command-line parameter to tell the program how many vectors you want. Sometimes, quantization is part of a preprocess, where we don't care much about speed. So, you can quantize your input vector just by comparing it against each of the dictionary vectors in sequence and outputting the one with the highest dot product. This technique is appealing because of its simplicity.

If you need more speed than that, you can BSP the set of dictionary vectors. To do this, first identify the neighboring vectors for each vector in the dictionary, and for each pair of vectors create a separating plane exactly halfway between them. Now construct a BSP tree from these separating planes. Mark each leaf node of the tree according to whichever dictionary vector belongs inside it. Now you're ready to encode very quickly.

## Rotations

We want to encode rotations in a manner that is isotropic in rotation space. Most ad hoc methods decompose the rotations into various parameters, such as Euler angles or an axis-angle representation, and then encode each coordinate separately. This process causes the same anisotropy problems we just saw with 3D unit vectors, but now they're more complex and harder to visualize.

We are going to encode rotations by leveraging the fact that quaternions are the natural, undistorted representation of 3D rotations. Justifying that statement is beyond the scope of this month's article, but I'll do it in a future column. For now, I'll just say that it's related to the fact that constrained quaternion interpolation gives you the minimal-torque transition between two rotations and refer you to the standard quaternion references on the web.

We could quantize the quaternion's *xyzw* components separately, taking into account that the fourth component of a unit quaternion is mostly determined by the other three. But this is not a good idea, since we want our quantization to be regular in curved space, not linear space.

The most bandwidth-efficient method is to treat the quaternion, a 4D unit vector, in exactly the way we just treated 3D unit vectors. We use a relaxation scheme to precompute a set of dictionary vectors, then match the input quaternion against that dictionary.

The biggest drawback with this method is that, because 4D space is big, we may need a lot of dictionary vectors to achieve the required resolution. If we have too many dictionary vectors, keeping a BSP tree of those vectors in RAM becomes cumbersome. We can start playing tricks to reduce this storage, for example by generating vectors that cover only 1/16th of the unit sphere and requiring the generated set of vectors to tile with itself to cover the sphere. The tiling constraint introduces a small amount of inefficiency, but it's not too bad. Try visualizing the equivalent trick in 3D space: imagine generating vectors for only one octant of the 3D unit sphere and attaching that octant to copies of itself in order to cover the whole sphere.

With quaternions, we shouldn't forget that we do not actually want to cover the whole sphere, because quaternions that face opposite directions represent the same rotation. So even if we generate vectors for a whole sphere, we then want to cull about half of them. If the input quaternion does not lie within the hemisphere we have covered, we try its opposite.

If you still need too much resolution for this approach to be practical, then I recommend using an axis-angle encoding, using the previously discussed 3D unit vector encoder for the axis and the scalar quantization for the angle. To make this method work best, you need to balance how much resolution goes into the angle quantization versus the vector quantization. I haven't worked this technique out, but the easiest thing to do would be to run a numerical optimizer that tries various possibilities until it achieves the best balance.

## Lastly

Often the vectors we want to transmit are biased somehow. For example, in an FPS or MMORPG, the direction a player looks is usually along the horizon.

You might take advantage of this situation by distributing the 3D unit dictionary vectors so that many of them point toward the equator of the sphere, and few of them point toward the poles. This trick allows you to achieve a low mean error for the directions that you transmit, but your maximum error will rise, which is probably not good. As an analogy, think of frame rate. If you are optimizing a rendering system to achieve high average frame rate, you don't want to do things that sometimes cause the frame rate to drop very low. Those drops will be very objectionable, even though the average remains high.

I think the best method of exploiting this kind of pattern is to keep the quantization size fixed as we did this month. Then, use a statistical encoding scheme, such as Huffman or arithmetic coding, on the output.

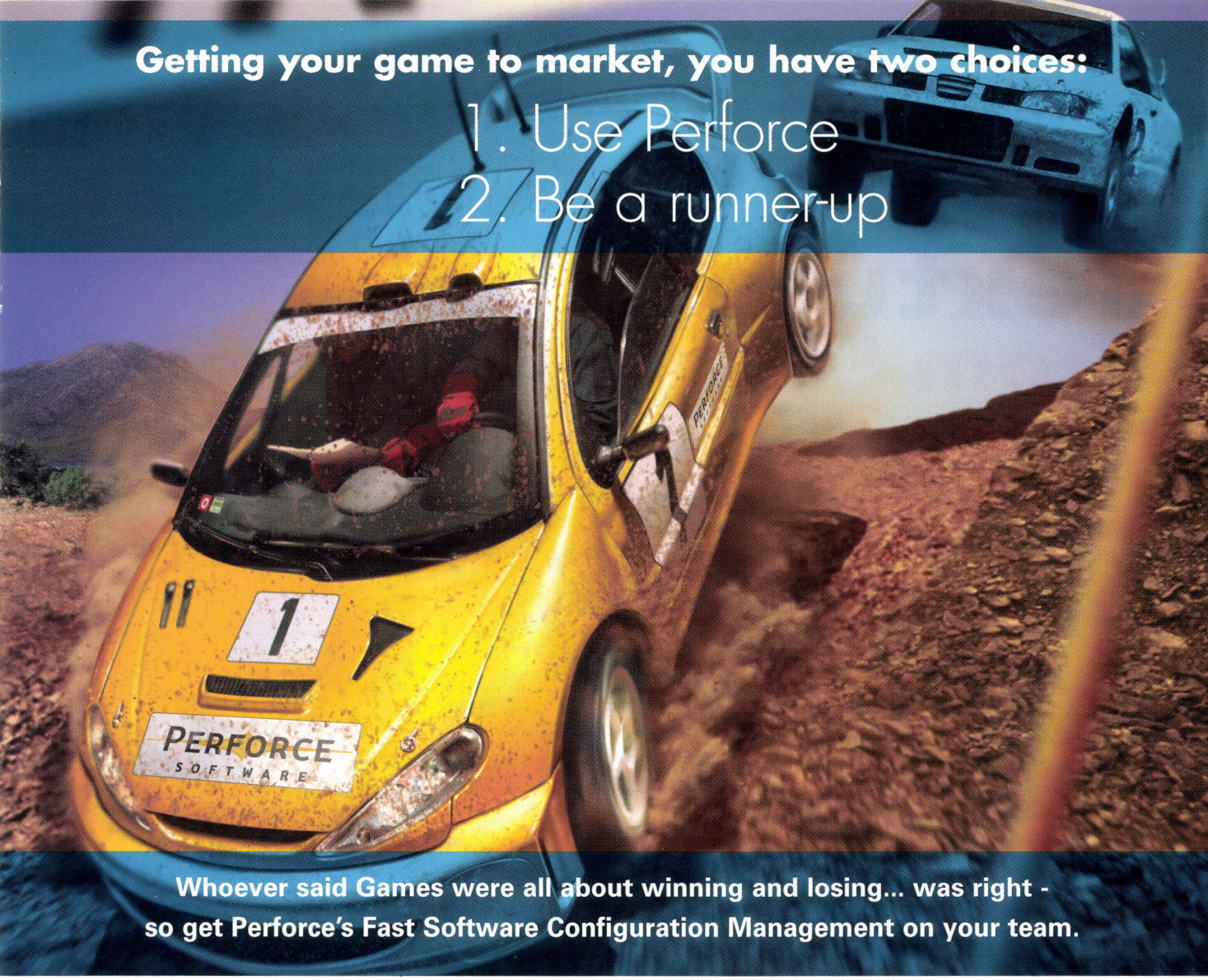
### FOR MORE INFORMATION

Rush, J. A. "Sphere Packing and Coding Theory." In *Handbook of Discrete and Computational Geometry*, edited by Jacob E. Goodman and Joseph O'Rourke, pp. 917–932. Boca Raton, Fla.: CRC Press, 1997.

Sayood, Khalid. *Introduction to Data Compression*, 2nd ed. San Diego: Morgan Kaufmann Publishers, 2000.

Frequently Asked Questions about Spheres
www.math.niu.edu/~rusin/known-math/index/spheres.html

Circle Packing
http://mathworld.wolfram.com/CirclePacking.html




**Perforce's Software Configuration Management System is the choice of winners, because no other tool can match the speed, control and reliability that it brings to the management of digital assets (binary and text files).**

***Maintain your top speed no matter how many users or files.*** At the core of Perforce lies a relational database with well-keyed tables, so simple operations can be accomplished in near-zero time. Larger operations (like labeling a release and branching) are translated into keyed data access, giving Perforce the scalability that big projects require.

***Install it fast, learn it fast, execute operations fast.*** With other SCM systems, developers face an unpleasant choice: do it the right way or do it the fast way. Perforce's speed and reliability mean the fast way is always the right way.

***Work anywhere.*** Perforce is efficient over high-latency networks such as WANs, the Internet and even low-speed dial-up connections. Requiring only TCP/IP, Perforce makes use of a well-tuned streaming message protocol for synchronising client workspace with server repository contents.

***Develop and maintain multiple codelines.*** Perforce Inter-File Branching™ lets you merge new features and apply fixes between codelines. Smart metadata keeps track of your evolving projects even while they develop in parallel.

***Supports all development platforms.*** Perforce runs on more than 50 operating systems, including all flavours of Windows, UNIX , Linux, Mac OS X and more.

***Integrate with leading IDEs and defect trackers:*** Visual Studio.NET®, Visual C++®, Visual Basic®, JBuilder®, CodeWarrior®, TeamTrack®, Bugzilla™, ControlCenter®, DevTrack® packages, and more.

PERFORCE
SOFTWARE

*Fast Software Configuration Management www.perforce.com*

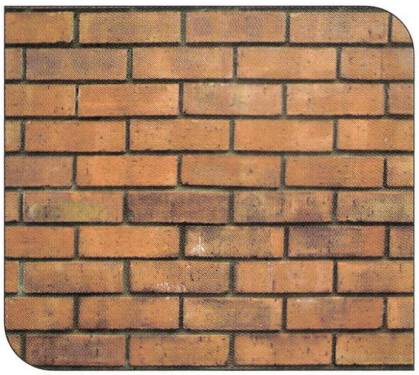
**Download your free 2-user, non-expiring, full-featured copy now from www.perforce.com**
**Free (and friendly) technical support is on hand to answer any and all evaluation questions.**

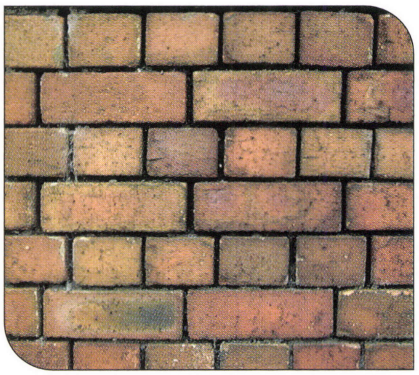
All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

# Textures
## From Source to Screen
## Part 3

In my past two columns, I've looked at types of textures and the processes involved in the transformation of source material into usable textures. This month, I'll look at some additional elements of this process, focusing primarily on tiling.

The texture-handling capacity of today's PCs and consoles is vastly superior to their counterparts from only two or three years ago. Yet as artists, we have yet to reach a point where all texture restraints are removed and we can happily crank out hundreds of unique textures at 512×512 and expect our hardware to display them all at once. The variation from console to console (or between graphics cards) is pretty significant, and while top-class texture compression or 128MB of on-board memory can in some cases give the illusion of freedom, overall texture capacity is still an issue. This is where tiling steps forward and offers its assistance.





FIGURES 1A & 1B (left and right). The uniform size and placement of the bricks in 1a lend themselves well to tiling. The irregular size and placement of the bricks in 1b will require additional artist labor to make them tileable.

## Find a Source That Will Tile

Talk to a grizzled veteran texture monkey and he may well tell you that he can make anything tile. This may be true in some cases, but the point is not really what can and can't be done — it's more about producing quality results in an acceptable time period. Thus, perhaps the first skill a prospective texture artist should acquire is the ability to identify images that will tile well and easily versus those that will require a team of 10 artists working through the night to pound them into shape. Consider Figures 1a and 1b as examples of good and bad tiles, respectively.

Both of these images represent fairly standard brick walls, but one image looks less tile-friendly than the other. The uniform size and regular placement of bricks in Figure 1a will ultimately make the task of tiling it far simpler than Figure 1b, which has two brick sizes and less uniformity in the way they are placed. There are many ways to accommodate problems like this, and the wall in Figure 1b may be exactly the kind you need. My point is this: don't make more work for yourself than necessary by beginning with an image that is awkward.

This is where the strategy of working with source material that is as high-resolution as possible pays off. While the image as a whole may be unsuitable, selecting an area that looks like it may work is entirely possible if the initial resolution is high enough.

## Know Where to Cut

For the sake of argument, let's limit ourselves to square textures. The same principles apply to other shapes (remembering to restrict their dimensions to powers of two), but it's simpler for illustrative purposes to use squares.

Choosing the part of the original image that you wish to work with is closely linked to the idea of finding a source that will tile. Obviously, selecting an area to use has a lot to do with the content of the image, as well as what the texture is ultimately supposed to achieve, but some factors to consider are:

- Distribution of color and brightness
- Variations in sharpness (are some areas more in focus than others?)
- Areas of strong shadow (which can

**HAYDEN DUVALL** | *Hayden started work in 1987, creating airbrushed artwork for the games industry. Over the next eight years, Hayden continued as a freelance artist and lectured in psychology at Perth College in Scotland. Hayden now lives in Bristol, England, with his wife, Leah, and their four children, where he is lead artist at Confounding Factor.*

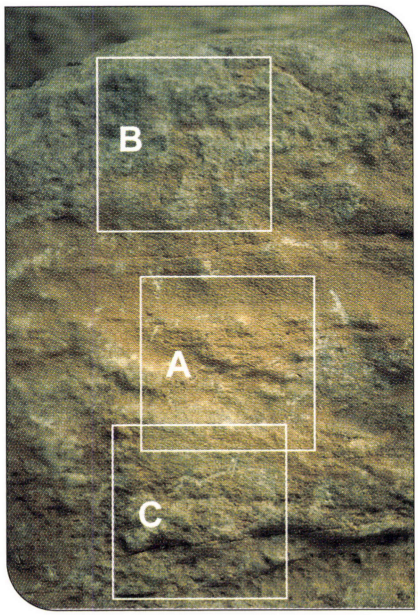



FIGURE 2. Evaluate the lay of the land before putting work into a tiling texture, taking into consideration variations in focus, color, and lighting.

make a texture too specific as to light direction).

In Figure 2, section A represents an area that would make a good starting point for a tiling texture. While it would still require some work, the color and brightness distribution is quite even — everything's in focus and the shadows in this area are not too harsh. Section B is not too bad in terms of color or brightness distribution but becomes distinctly out of focus toward the top. Section C moves out of focus toward the bottom but also has significant shadows within its boundaries, which would be time consuming to remove.

## Preparation

If you'll excuse me for stating the obvious, a texture that tiles correctly is one that you can repeat vertically and horizontally across a surface, without an obvious join between tiles. A texture that tiles well, however, is a little more than that. Simply disguising the join between the edges of a tile may be enough if the repeat is going to be very limited, but with a texture that has more visible repeats, you must take care with the entire texture, not just its edges.

First, consider the fact that the bottom and top edges of the texture will need to match, as will the left and right edges. The following section will deal with the majority of this process, but in the preparation phase, we need to look at things such as matching color and brightness edge to edge and, where possible, evening it out across the texture.

There are many ways to do this, but one useful method uses the Quick Mask feature in Photoshop. The original image (Figure 3a) darkens off significantly toward the bottom and is obviously not going to tile correctly. By using a simple black-to-white linear gradient, as Figure 3b shows, in conjunction with the Quick Mask feature, you can make a selection that incorporates the values within the gradient to allow a more controllable "feathering" effect at the edge of the selection.

In other words, where the gradient is 100 percent white, no pixels will be selected, and where the gradient is 100 percent black, all pixels will be selected. However, the selection also covers the shades of gray in between. Thus, whatever effect you then apply to the selection will be moderated between the maximum and minimum values according to the distribution of the gradient.

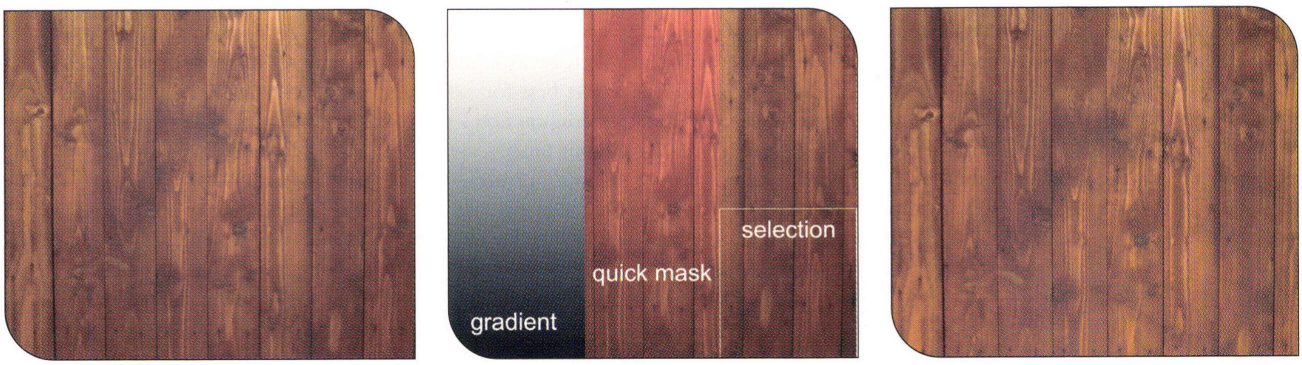
Using this method, you can apply any operation (level adjustment, color balance, and so on) across the area of the texture that needs altering and blend out smoothly to the areas that need to remain undisturbed.

To some extent, you can use this same method with the Sharpen and Blur filters to compensate for areas that are focused unevenly. This technique has limited value, however, as a sharpened image is artificially adapted to provide the appearance of crisper detail and doesn't actually refocus. Too much sharpening in certain areas will leave a horrible, pixellated mess.

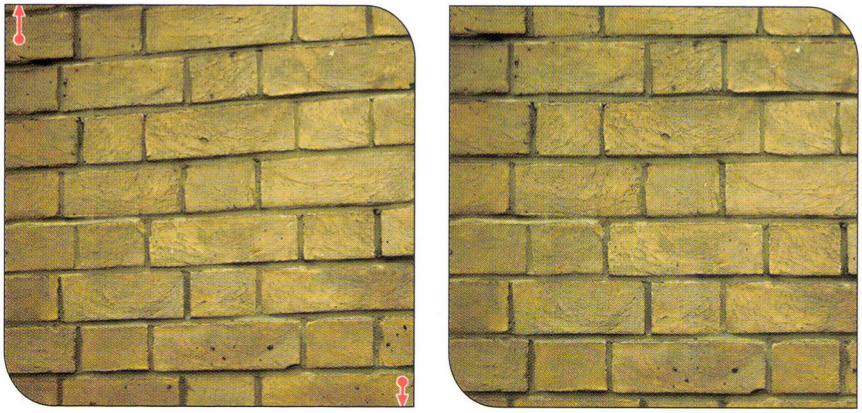
It's also worth noting that you can use a radial gradient if the adjustments needed are at either the center or the edges of the texture.

## Perspective Problems

Many textures will, by their nature, contain features that run parallel




FIGURES 3A–C (from left to right). The lighting problems in the original image (3a) can be corrected with a white linear gradient and Photoshop's Quick Mask feature (3b), resulting in a more readily tileable image (3c).



FIGURES 4A (left) & 4B (right). Fixing a faulty perspective in a source image (4a) with Distort to create a tileable result (4b).

to each other. As I discussed back in May in part 1 of this series, it's always best to use an image that was taken at a 90-degree angle to the surface in question, to minimize any warping that occurs due to perspective. Still, in most cases, your source material is not going to be perfect, and you will need to correct perspective problems before a texture can be tiled.

Again using Photoshop as an example, you can apply the Distort function to adjust the problem image. Brick walls are a good example of a texture where both vertical and horizontal lines come into play. Figure 4a shows how you'll need to stretch the original image in two directions so that the horizontal lines between the rows of bricks run level with the top and bottom edges of the texture (Figure 4b), hopefully lining up at the left and right edges.

Photoshop's Distort function allows you to move each corner of the image independently (outside the boundaries of the image if necessary), which gives you the ability to realign the edges. The quality of the resampling means that loss of detail is kept to a minimum, and once again, using a higher resolution than will ultimately be necessary makes this largely irrelevant.

## The Basic Maneuver

Once you're satisfied that the texture is ready to be tiled, it's time to get your hands dirty. There are a variety of ways to tile a texture, and some applications will even claim to do it for you (with varying degrees of success), but the basic method is about as straightforward as you can get.

First, offset the texture so that the edges (horizontal and vertical) now run across the middle of the image. In Photoshop, the option needs to be set to Wrap Around. The offset doesn't have to bring the edges exactly into the center, but it helps to rearrange the image so that what are now the edges, having previously been the center, automatically tile.

Next, cover the joins that cross the center of the texture. This is usually done either by cutting and pasting sections from elsewhere on the texture or from a similar image, or by using the Rubber Stamp tool to duplicate other areas across the join. Using areas from within the texture itself to cover the join can lead to noticeable repetition of features. While the Rubber Stamp tool is quicker, it is also less easily controlled because the brush shape is fixed. It works best for smaller areas, especially in conjunction with a pressure-sensitive input tablet.

Cutting and pasting an area can provide a custom-shaped patch that you can tailor to fit in with the texture underneath. Setting the selection tool to feather automatically will blend the edges in slightly, but the important work is best done by hand, by erasing the edges through to the layer beneath in an attempt to merge them together seamlessly. Again, a tablet makes this job much easier. Certain surfaces, such as a stone wall, require careful selection of which pieces to cut and paste, as you will have to make them fit in with the layout of the stones around them.

Finally, once you've covered the joins, offset the image again. Doing so allows you to deal more effectively with the parts of the joins that were right at the edge of the texture. The texture should now tile seamlessly in both directions.

## Balancing Act

In the course of our tiling preparations, you should have removed large-scale variations in brightness or hue so that the texture would match horizontally and vertically. On a smaller scale, you need to distribute (or reduce) features within the texture to minimize obvious repetition, or patterning.

You should do this, in part, during the actual tiling step. Cloning and cutting and pasting should take into account the overall color and brightness distribution within a texture and aim to balance them out. It's unlikely, however, that the process of covering up the joins as I just described will be sufficient, so you should assess the texture further to see if more work is necessary.

This balancing act is not just about controlling the distribution of features in the texture — it's also about deciding how much to homogenize elements in a texture for the sake of minimizing visible repetition. Enforcing too much homogeneity can destroy the detail that makes a texture good in the first place.

The amount of repetition that each texture is likely to be subjected to (grass may stretch on for miles, whereas wood is generally localized), gives some indication of the demands on each texture. But, as always, the choice is up to you.

Tiled textures have long been a staple of the videogame artist. Even as the power of our game machines explodes through the roof, creating quality textures that tile effectively remains an important skill, one that is certainly worth refining.

# FACIAL ANIMATION STRESSING YOU OUT?

## You Need Eyematic® FaceStation™

The World's First Intelligent Facial Animation System™

Got a briefcase full of money for proprietary motion capture hardware and software? Got a team of highly-skilled motion capture engineers? Got an army of animators to tweak keyframes? Got any aspirin?

Eyematic puts an end to the madness - with FaceStation, you no longer have to be rich or famous to create high-quality facial animation. Eyematic's patented computer vision software automatically locates and tracks facial expressions and mouth movements in real-time, using just a single video camera or camcorder. No awkward facial markers, no time consuming post-processing and no expensive dedicated hardware. Just an integrated suite of software applications that lets anyone easily create expressive, realistic facial animation.

Visit our web site to find out how FaceStation can add the power of professional facial animation to your tool chest.

www.eyematic.com/face

Eyematic FaceStation
Only U.S. **$1,995**

Version 1.5 Now Shipping for Discreet™ 3ds max™ Users
Alias | Wavefront® Maya® Version Available Summer 2002

EYEMATIC®

Copyright © 2002 Eyematic Interfaces, Inc. Pricing is suggested retail pricing only, and may also vary by country. Eyematic is a registered trademark and FaceStation and Intelligent Facial Animation System are trademarks of Eyematic Interfaces, Inc., but you could probably guess that. Other product or service names mentioned herein are the trademarks of their respective owners.

# Audio Tools: You Get What You Ask For

Creating all your sound effects and music for a game is one thing; getting them actually installed into the game is quite another. Handing over the audio to be installed to a programmer — who may or may not have any kind of audio sensibility — is one approach. Getting that programmer to do what he or she does best — coding — and create tools that allow you to implement the audio yourself is another option. The more flexible the tool, the more power you have to get the game sounding right. But with flexibility comes complexity, and at some point a tool becomes so flexible (and thus complex) you'd have to be a programmer to use it. The best tools strike a critical balance and leverage the talent on the team — audio people making audio, programmers programming.

**Do your homework.** You won't get anywhere specifying tools that are impossible given the game's basic architecture, platform, and other considerations. So make sure you understand all of these parameters first. It's also a good idea to figure out what tools have been created for other disciplines that audio can easily piggyback on. You can get great mileage out of simple audio extensions to the animation, level building, or cinematics tools.

For example, the level editor might specify surface material, the animation system might keep track of when a character's foot hits the ground, and the physics system might know how hard it hits. With tools providing simple access into these systems, you can play the correct footstep sound at the correct time with the correct volume and pitch parameters. Such a system also provides other potential benefits: improvements and refinements to these tools will likely improve and refine the audio implementation as well.

**Know what you want.** Once you know what's possible, figure out what you want. It sounds obvious, but you can't get what you want until you know what that is. Brainstorm, invent sample implementations, test edge cases, consider every wild contingency — make sure that your design really does solve the problem, and most importantly, that it's solving the right problem. Programmers hate nothing more than trying to solve a changing, poorly defined problem. Such planning may seem counterintuitive in the creative arts, but in fact, if you haven't done this level of due diligence, you're doing yourself and your game a disservice anyway. Be prepared, be thorough, and be right the first time.

Get your terminology together early, too. If you go into the process referring to a unique sound as a "sound effect" and your programmer refers to it as a "one-shot," you are going to have a problem. Programmers will usually be very flexible if you establish terminology up front and then use it consistently. Also keep in mind how your work is categorized — by functionality, by usage, by game area, and so on. It may make sense to you to organize your sounds together by their location in the game, whereas to the programmer, categories based on functionality see more logical. There is no one right way; just agree on whatever works up front and be consistent until the end, or prepare for an organizational nightmare

**Get it in writing.** Once you know what you want and what you can have, create two types of documents to communicate your needs. First, distribute a global "Basic Functionality" document to the audio programmer, the lead engineer, and any designers, producers, art leads, or others who are interested. It's structured as a bulleted list in simple prose, ordered from most basic to more sophisticated. In general, this document is bone simple — use a template that, with only minor modifications, is applicable to almost every project on almost every platform. It actually starts with the requirement that the game be able to identify a sound, play a sound, and stop a sound. The point of this document is to ease everyone into thinking about audio and to lay an important functional foundation.

For the details, write a full Functional Spec for each tool. The Functional Spec should be very, very thorough, stating the problem the tool is designed to solve, describing in extreme detail how the solution works, and then providing one or more sample usages. It also describes what kinds of errors the tool should report and how it should behave in every edge case, and makes predictions about likely bugs. As complete and thoughtful as this document is, it will go through a number of revisions before you are finished with it. Keeping the document and the tool in sync is worth it: if the Functional Spec is well made, you can be sure that everyone who reads it will understand what the desired result is, how the result is to be achieved, and if it's not working that way, a good idea of why not. Your game will sound better for it.

**ANDREW BOYD** | *Andrew is audio director at Stormfront Studios. He is currently audio directing* The Lord of the Rings: The Two Towers *for PS2. Other recent projects have included* Blood Wake *on Xbox and* Pool of Radiance: The Ruins of Myth Drannor *on PC. Drop him a line at aboyd@stormfront.com.*

# FORGET EVERYTHING YOU'VE HEARD.

SOUNDMAX SPX

Someday, you'll look back and remember the moment sound changed.

Enter SoundMAX SPX, powered by the SoundMAX Audio Rendering Technology Toolkit (SMartTools). Create ultimate, immersive audio, using advanced audio algorithms to deliver hyper-realistic game experiences.

We've blown through the sound barrier. And warped into the next generation of sound dynamics. Now you can sculpt, manipulate and control complex audio behaviors. Form real-time sound events that you animate on-the-fly as the game is played.

Exploit the power of SoundMAX in your next project. Fully scalable. Cross platform. It's your choice. Get SMart. Sound better.

Remember where you heard it first.

WWW.AUDIOFORGAMES.COM

ANALOG DEVICES

SoundMAX® is a registered trademark of Analog Devices, Inc.

# Turn-offs

*This month we consider a rule from Dale Geist and the LucasArts Game Theory Group, by way of Hal Barwood and my own editing. I'm also posing a game design challenge: Name That Trump!*

Naughty Dog's JAK & DEXTER gives players a sense of freedom and adventure.

**The Rule: Let players turn the game off.**

A player should be able to save and exit the game at any point, losing at most a few seconds of progress as a result. Our objective as designers is to entertain, not punish — and many games force players to play for extra minutes, even hours, until they can reach a save-game point. Such a system forces players to recapitulate those minutes if they quit prematurely, in frustrating repetition of now-familiar events. It's a commercially important rule, akin to the old adage "The customer is always right." Players have been known to give up on games that did not follow this rule, and even return them.

**The Rule's domain.** This rule applies to all single-player game genres — or does it? See The Challenge below. This rule does not apply to most multiplayer games, because one player's right to turn the game off at any point is another's ruined game.

**Rules that it trumps.** See The Challenge below.

**Rules that it is trumped by.** This rule is trumped by the rule "Protect the player's suspension of disbelief." As I discussed in May's column, we don't want players to be reminded, "It's only a game." Beware of this rule if providing the ability to save and quit at any time makes it easy for the player to subvert the game-world experience by applying the real-world trick of saving with each incremental advance and restoring with each incremental failure. For example, in an RPG, it may be better to avoid saving during combat, to discourage the player from taking a swing and saving if it connects or restoring to the last save if it doesn't.

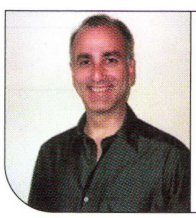
**Examples and counterexamples.** JAK & DAXTER allows you to save anywhere, and in return you are rewarded with a remarkable sense of freedom. RAYMAN lets you save only between levels, and many other games, such as the recent HARRY POTTER AND THE SORCERER'S STONE game, require you to save only at certain points, designated by a special icon or object in the game world. These games can subject the player to a frustrating repetition of long gameplay sequences when it becomes necessary to shut the game down between these points.

**The Challenge.** Name That Trump! As experienced players have probably already noticed, there are many games out there — some very popular — that don't follow this rule. And yet the rule seems logical. What's the catch? What other rules that enhance the gameplay trump this rule and dictate that the player must finish a level or a section of a level to save? I can think of at least two possible rules that would trump or otherwise contradict this one under certain circumstances, but I suspect there are more.

I'm challenging the readers of this column to submit these counter-rules that trump this one or are trumped by it. Solve the mystery of the missing between-level save-games. Here's one hint: the freedom to quit at any point can drain a game of dramatic tension. Freedom from anxiety is good. Freedom from excitement is boring.

**Rules: They're not just the law. They're a good idea.**

I'd like to conclude with an observation. As I've struggled to articulate game rules — and it often can be arduous — I've discovered something unexpected. Some rules I thought were trivially simple prove quite complex when set down in print. Others that I thought were inherently obvious turn out to have hidden contradictions or inconsistencies when I tried to write them up. But the main point is that the actual process of trying to articulate a rule has been a valuable learning process for me. I urge you to try it yourself, even if you don't send your rule in. You might just teach yourself something new.



**NOAH FALSTEIN** | *Noah is a 22-year veteran of the game industry. You can find a list of his credits and other information at www.theinspiracy.com. If you're an experienced game designer interested in contributing to The 400 Project, please e-mail Noah at noah@theinspiracy.com (include your game design background) for more information about how to submit rules.*

# GameDevelopers Conference Europe

27-29 August, 2002
Earls Court Conference Centre • London, UK
**www.gdc-europe.com**

# Make better games

The Game Developers Conference Europe 2002 focuses on real world problems and solutions. Sessions allow the exchange of hard earned knowledge and practices between the game industry's most successful developers.

**Topics include**

- Build or Buy: The Muddle of Middleware
- Where is Technology Going? A Panel Discussion
- Critical Analysis of Games: Panel & Debate
- Ratings and Censorship: Making Sense of Videogame Ratings
- Developing in Developing Countries
- Publisher Developer Relations: Bridging the Gap
- Postmortems on:
  - Scripting in BLACK & WHITE
  - AI in STATE OF EMERGENCY
  - Production of PROJECT GOTHAM
  - Audio in THE GETAWAY

**Keynote**

**Preproduction: Myth vs Method**
**Mark Cerny** Cerny Games

**Speakers include**

**Simon & Dene Carter** Big Blue Box
**Sarah Chudley** Bizarre Creations
**Charles Cecil** Revolution Studios
**David Braben** Frontier Developments
**Francois Masciopinto** Infogrames
**Glenn Corpes & Jeremy Longley** Lost Toys
**Jonny Dobson** VIS Entertainment
**Jonty Barnes & Richard Evans** Lionhead Studios
**Marek Spanel** Bohemia Intertactive
**Phil Harrison** Sony Computer Entertainment Europe

**Sponsored tutorials include**

- ATI Mojo Day
- IGDA Academic Day
- Meltdown UK: Microsoft DirectX Developer Day
- Xbox Unplugged

**For a complete listing of topics and speakers, visit www.gdc-europe.com**

in partnership with ects 2002 29-31 august earls court london

steven.woodcock

# Game AI: The State of the Industry 2001–2002

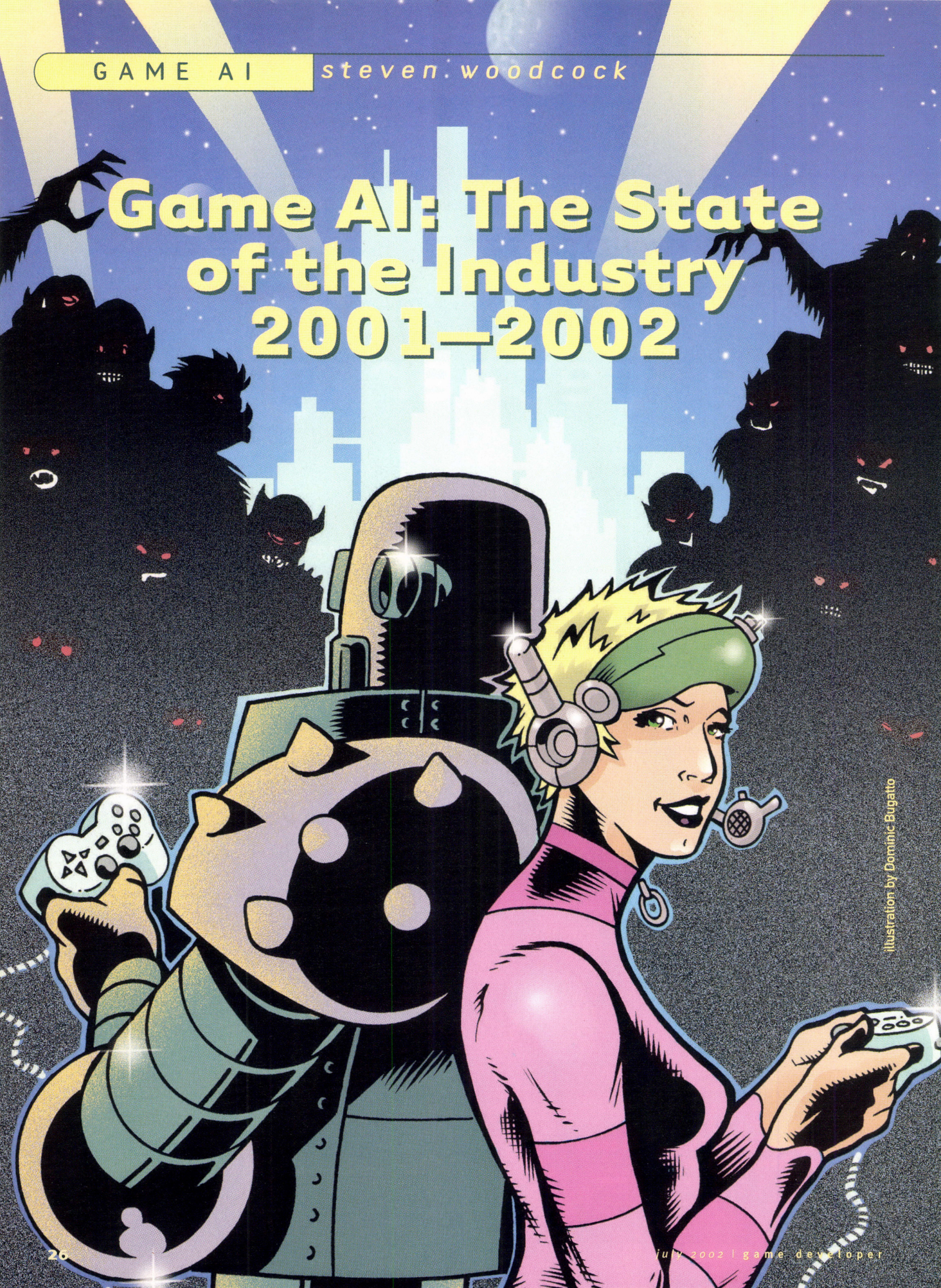


illustration by Dominic Bugatto

# Focus Focus Focus

Last year's State of the Industry report ("Game AI: The State of the Industry, 2000–2001," August 2001), indicated that developers had finally gotten good game AI to be taken seriously. AI developers were part of the integral design process and were turning to tried-and-true engineering techniques to build and modify their AIs. Rather than focusing on flashy new features or technologies that didn't quite pan out, they were focused on one thing — building their AIs with a more straightforward and elegant approach.

After attending the 2002 Game Developers Conference (GDC) in March, I was amazed to see how these trends are accelerating. As one might expect, the focus on techniques over technology is continuing. Developers are in a serious learning mode and seeking better ways to do the basics — pathfinding, obstacle avoidance, planning. They already have two or three games under their belts and lots of code to lay their hands on; what they want now are new ideas, ways to do things better and faster, and ideas to make their AIs look smarter with less effort. In short, they're where the graphics folks were at the dawn of DirectX — hungry for more.

Developers are still being inspired by the designs of others, though their focus is more broad-based than in the past. Last year most developers cited Maxis's THE SIMS as their primary inspiration, looking at its design as the basis for many other types of games. This year, while THE SIMS still plays a major role as an example of "AI done right," many other games were cited as having excellent AI design. Poptop's TROPICO in particular came up as an excellent example of a very focused and effective design, as did Timegate Studios' KOHAN: IMMORTAL SOVEREIGNS and its sequel, AHRIMAN'S GIFT.

Developers want to make that kind of focus a part of how they do their job; it was a recurring theme throughout the Conference.

## Those Ever-Expanding Resources

It's amazing how much better off developers have it in the current crop of designs than they had in the past. This feeling of well-being didn't seem to match up with the results of our annual basic resources survey, however. For the first time at the GDC AI roundtables, the raw numbers (Figure 1) indicate that developers actually lost some resources overall.

Does this data paint a grim picture indicating that, in the wake of corporate downsizing and the collapse of the tech sector, AI developers' lives are about to get harder? No, not really.

The resultant discussion of these numbers was fascinating. Most developers feel that, when it comes to resources, they have it pretty good. While they all agreed that they have slightly smaller teams and slightly less CPU resources overall, they also didn't feel as if they needed as much as they had in the past. For their part, the turn-based strategy game folks reported both more available CPU (100 percent if they needed) and more dedicated AI developers on average (anywhere from 1 to 2.5 developers) than developers of other kinds of games did.

The reasoning for this turn of events ran the gamut from better tools to more experience to just plain bigger CPUs on the target platforms. Developers have built themselves all kinds of toolkits to help them build game AIs, using scripting and libraries from previous projects more efficiently than before. Experience with previous games helped developers in optimizing the last game's AI libraries for the next. (The wave of sequels we saw in 2001 also encouraged this somewhat.)

Developers also felt that their target machines just kept getting faster and faster. Target hardware featured faster CPUs and better graphics cards than in previous years, providing more cycles than before. Even the console developers reported that they were far more constrained by a lack of memory and storage than anything else.

It is interesting to note that some developers were reporting their CPU usage in milliseconds rather than in percentage terms. That's really a better measure and reflects a number of interesting trends, including an increased use of developmental resource tools that provide these kinds of measurements.

## Development Now: Focused on the Tried and True

It's clear from the roundtables and isolated discussions at GDC that AI developers are much more focused on refining their current designs and engines than on innovation. They want to take what they are doing and make it better, more efficient, and more lifelike. They are spending more time on design and thinking about how to use the tools they've already developed than on researching new approaches. Developers want to focus on results — faster execution and less memory are the order of the day.

**Doing more.** It was fascinating to see how developers are accomplishing these goals, since it might seem at first glance that they are resting on their laurels. After all, if they are only refining their

**STEVEN WOODCOCK** | *Steve's background in game AI comes from 18 years of ballistic missile defense work building massive real-time war games and simulators. He did a stint in the consumer arena, then returned to the defense world to help develop the AI for the national missile defense system. He maintains a web page dedicated to game AI at www.gameai.com and is the author of a number of papers and publications on the subject. He now pursues game AI through a variety of contract work and served as contributor to and technical editor for several books in the field, including the* Game Programming Gems *series and* AI Game Programming Wisdom. *He can be reached at ferretman@gameai.com.*

tools and AI engines to make sure the next sequel acts a little smarter, they're not really taking too many risks or learning much new, are they?

As it turns out, nothing could be further from the truth. Developers reported that after tweaking their engines to do things more efficiently than before, they are using that extra horsepower to just plain do more. They're building their engines to do more line-of-sight checks per AI cycle, to allow for deeper pathfinding for the player's units (so they don't look so dumb), or to let more bad guys run through more decisions than before. First-person shooters, for example, might have previously focused on having only the three enemies closest to the player's character "act smart" and get lots of CPU cycles so they'd be intelligent enough to dive out of the way of a grenade. Now that developers have a handle on making their engines work more efficiently, they're using those extra CPU cycles to include not only the three enemies close to the player but also their five comrades behind them in the second squad.

Developers have also realized that in many games their AIs really don't have to be smart for very long. The primary purpose of game AI is not necessarily to crush the player mercilessly but to entertain the player and provide a challenging opponent. The lifespan of most tactical enemy AIs is relatively short — they live to die, essentially. This means that they only have to look smart, on average, for five seconds or so; they don't have to be any smarter than they need to be. This understanding is driving developers to add a little bit more smarts to their AIs, and as a result they're more realistic than ever before.

Westwood's COMMAND & CONQUER RENEGADE, uses a real-time analysis capability allowing the game to learn from a player by adding pathfinding nodes to its list of valid paths when the player does something novel, such as jumping out a window.

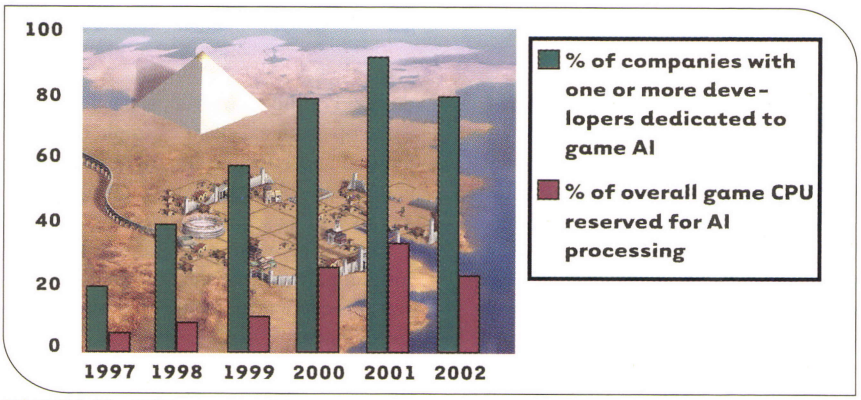



FIGURE 1. The 2002 resource picture.

One session, hosted by fellow AI moderator Neil Kirby, provided an excellent example of this kind of realism in Red Storm's GHOST RECON. In that game, the enemies have a number of "idle" animations, including a fidgeting-like behavior in which the bad guy glances around and shifts nervously. Since he can look around, he stands a chance of glancing at the player who might be trying to sneak up on him. Previous games were notorious for instances where an enemy completely ignored the player under these circumstances, since the player wasn't in the AI's activation zone. In this case, however, the AI developers modified their AI's perception code to actually react to the player when this happened.

The result is a bad guy that reacts much more realistically, making the player work harder without actually doing anything else much different. (Another interesting point is that this particular enhancement came originally from the game's art team, not its AI team, again showing that game AI is something that all developers are taking more and more for granted as a feature of a good game.)

In one session, developers also noted that game AI tends to get better in stair-step fashion, as more effort and cycles are added. These stairs are particularly even, however, and the curve of AI smarts versus the developer's programming effort is nothing like a simple straight line. It takes care and experience to judge when "enough is enough."

**Languages and scripting.** Developers have been doing some interesting work with languages and scripting that shows they're aiming toward greater modularity and efficiency. Most developers are still using C or C++ as their primary language of choice, though there are a few scattered exceptions. The AIs in Bungie's HALO and Naughty Dog's JAK & DAXTER both make heavy use of Lisp and scripting. A few games (such as the DEER HUNTER–style games) continue to use Visual Basic, though they're a rare bunch.

Scripting itself is still seen primarily as a tool to save development time and let AI folks focus on the AI's features above all else. Several roundtables at this year's GDC focused on scripting, and it was obvious that a wide variety of scripting languages are in use across the game industry. Developers are using everything from Perl and Lisp derivatives to the more exotic Lua or home-rolled scripts. Tweaking the scripting engine from the previous game before diving into the next is common.

As a result of the scripting trends there have been more games released in the past year that allow a talented player to modify the AI's scripts and units to some degree. Here the player benefits from a decision by the AI developer to get out of the level design business, and the AI

developer benefits in turn from seeing a huge variety of new AIs and ideas for improvements from the user community. As in previous years, a good scripting engine also helps get the entire production done with fewer people (another reason why teams have shrunk slightly).

**Assimilating technology.** The packed roundtables showed that developers are still asking a lot of questions about new technologies that might be useful. It was clear that they are going to continue to adapt new technologies to their AIs very slowly as compared to their academic brethren.

In general, the AI roundtables showed that if anybody was using any "exotic" technologies, they just weren't talking about it. All the developers present said they were continuing to use straightforward finite state machines and fuzzy state machines to build their AIs. These were combined with scripting to handle missions and the general personalities of the computer opponents. A couple of developers spoke at length about how they were considering using neural networks for racing games to control the computer cars, but they weren't encouraged by others who'd had already dabbled in them.

Most developers seem to be using (or have built) layered AIs in their products, usually breaking out decision making along strategic and tactical levels. The strategic-level AI seemed to get the broader problems: deciding general strategies, performing map analysis, setting goals, and managing resources. The tactical-level AI got the "grunt" jobs: pathfinding, handling formations, and fighting. Interestingly, some developers thought that computer opponent behaviors were a strategic-level function, while others pushed it down to the tactical AIs. The first group argued that it affected goals and resource management, while the other group said it mattered more in formations and combat. While most developers are using two levels when they layer their AI, the developers of Stainless Steel's EMPIRE EARTH spoke of using up to seven layers for their game.

The developers of KOHAN noted that their AI engine gives more CPU time to the AIs at higher difficulty levels so players can consider more strategic options more deeply.

Terrain analysis came up as being new to many AIs and eating many of the CPU cycles freed by improvements in other processing. Most games are doing it in one fashion or another, though few are spending enormous amounts of CPU on it. Most developers use some kind of terrain analysis either to identify good paths prior to the scenario or to update valid paths as the game progresses. Terrain analysis helps most strategic-level AIs in identifying locations of resources or chokepoints on the map and is generally done during the "idle" moments when the AI isn't over-tasked with doing anything else.

A few developers are using cycles to analyze maps for connectivity and building portals or waypoints for later use by their pathfinders. One game, Westwood's COMMAND & CONQUER RENEGADE, uses a real-time analysis capability allowing the game to learn from a player by adding pathfinding nodes to its list of valid paths when the player does something novel, such as jumping out a window.

Compared to previous years' discussions, there really wasn't much emphasis on pathfinding issues this time around. What discussions there were focused less on how to find the best or shortest path and more on ways to move units more intelligently. There was general agreement that all games should be designed to use hierarchical pathfinding, with one pathfinding for high-level paths and one for lower-level obstacle-avoidance issues (both Blizzard's STARCRAFT and Ensemble's AGE OF KINGS do this, for example). Everybody was interested in ways to minimize the CPU hit of pathfinding and do it only when absolutely necessary.

All of this tied in well with the general trend that AI developers are focusing more on learning the tools at hand, and using them better, than on exploring new technologies. That doesn't mean, however, that developers have abandoned all experimentation.

## Development Next: Focused on Possibilities

Developers at GDC talked about a lot of possibilities for their future projects, some of which seemed at odds with their current mode of retrenching and regrouping. Nevertheless, it was interesting to kick around ideas about what they might look at next.

**Focus on learning.** If there was one thing to take away from the roundtables regarding what developers are aiming to do for future projects, it was that they are focused on building AIs that can learn to some degree. The problem is that while nobody thinks this is very easy, everybody wants to figure out ways to make it happen. The learning capabilities of Lionhead's BLACK & WHITE arose as something to strive toward, although there was general agreement that, for most games, learning on the part of the AI has to be more "behind the scenes" than it was in BLACK & WHITE. Many agreed that learning counts for a lot when it's done right, but most developers reported that they don't bother with it outside of limited use during development proper, although the C&C RENEGADE example prompted a flurry of note taking by many.

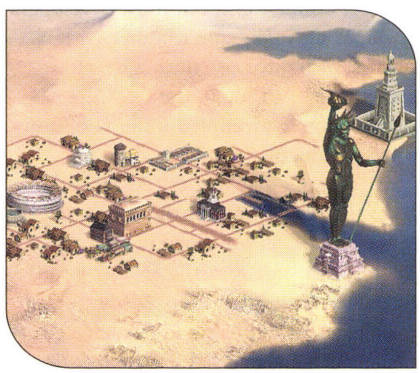
Specifically teaching the AI to learn from the player or over the course of many games, as was done in BLACK & WHITE, just isn't something most developers are bothering to do. Most seemed to think it would be dandy to have an AI that could ratchet itself up to meet the ability of the player, but nobody had a clear idea on how to keep the AI from learning something dumb (just like a player). Some developers built on ideas

from previous years to use limited learning during development and play-testing, generally to tweak existing parameters over the course of hundreds of runs. However, these parameters are always locked down before the game ships.

Overall, learning is seen as a logical extension of the current "do more with what you have" philosophy. It's viewed very much as a potential discriminator for future games, and we should see some interesting developments along these lines.

**Focus on scalability.** The current focus on layered AIs lends itself naturally to the next step, AIs that can easily scale depending on difficulty level and available CPU cycles. Some of this has already been done, though most games so far approach difficulty levels in a very straightforward manner — they either give the smarter AIs more resources at the start of the game (very common) or they let the smarter AIs cheat by viewing more information (ignoring fog of war, for example).

Many developers are experimenting with having a game loop perform ongoing terrain analysis whenever there are spare CPU cycles, so that the AI slowly gets smarter about the map in a manner similar to the player as he or she explores it. Others are making their AI's CPU cycles more elastic than they had been in the past now that they're generally free from the tyranny of sharing the CPU with the graphics engine. These designs are able to use anywhere from 5 to 60 percent of the machine's CPU cycles if necessary. The developers of KOHAN, for example, noted that their AI engine gives more CPU time to the AIs at higher difficulty levels so they can consider more strategic options more deeply.



FIRAXIS'S CIVILIZATION III allows players to view basic diplomatic relations on a big chart but gives them limited tools to make proposals themselves.

A few developers said that they are moving away from multiple levels of difficulty in future designs, as their feedback indicated that there are only two modes players really care about for solo play — easy and hard. Anything else is done online. It will be interesting to see whether this is a long-term trend that will affect future AI development or just a response to the rise of online games.

**Focus on diplomacy.** It's been said before, but diplomatic AIs really are becoming something that developers are viewing as more significant, particularly in multiplayer games. Players are tired of AIs that are just plain dumb, and this in turn has become a big driver in how players treat AIs in multiplayer games. One developer noted that people act differently in real life from how they act in games; in real life people tend to follow a leader, while in games they tend to gang up on him.

This makes some aspects of diplomacy very difficult, especially considering that many multiplayer games start out with a general "let's gang up on the AI" agreement among the players. Developers are exploring ways to short-circuit this tendency and get human players to consider allying themselves with an AI player. The problem is that it's not easy for an AI to communicate, coordinate, or even set mutual goals with the player.

Communication between the AI and the player might seem a simple task, but even that is difficult. Firaxis's CIVILIZATION III, for example, allows players to view basic diplomatic relations on a big chart but gives them limited tools to make proposals themselves. Some games, such as KOHAN: AHRIMAN'S GIFT, have crude facilities for players to "tell" an AI something, but it's difficult for them to judge how the AI responds to it.

Since much of the art of diplomacy lies in the subtle give and take of negotiation between players, developers want to build games that provide more than the bare-bones take-it-or-leave-it options common today. One roundtable featured a wide-ranging discussion on the merits of Nash's theorem (a gameplay theorem for optimizing moves based on perception of diplomatic moves with imperfect communication), though it was clear that this goes far beyond the problems most developers are worried about.

Beyond the problem of getting the player to engage the AI, the AI needs to communicate the other way as well, with the player. Building a strategic AI that recognizes and can communicate effectively the coincidence of interests between itself and a player is very difficult, but most developers think it is essential if they are ever to build an AI that truly plays more like another person. Building AIs that can potentially propose mutual goals for itself and the player is extremely important if AIs are to be considered seriously for an in-game alliance. We've all seen games in which the AIs seem possessed by schizophrenia, offering a peace treaty one moment and demanding tribute the next. One developer pointed out that there are few games that provide facilities for a diplomatic AI to work with the player to set a goal for their alliance, such as "You attack there and I'll defend our side."

Coordination between players and AIs is another area of potential future enhancement. Developers recognize that most AIs have a hard enough time getting their own units to work together, much less interfacing with the players or understanding the player's intent. Some games try to work around this problem by letting allies see each other's units (such as in AGE OF KINGS), while others even let players take over their units completely. A couple of developers plan to let one player "draw" on the strategic map to indicate proposed goals or attack sites, and most developers thought that getting the AI to do this itself would be a big step.

In all cases, developers plan to experiment with various methods of improving the user interface to the AIs in order to try to make their diplomatic AIs stronger.

The turn-based strategy game developers will probably lead the way here, since they have more CPU cycles available, but the real-time strategy genre is also ripe for developments.

## The Middleware Issue: Better AIs?

First broached a couple of years back in some of the GDC roundtables, the subject of AI toolkits or SDKs, now more commonly called middleware, was again a major topic at this year's GDC. Developers are very curious about what kinds of AI middleware are available and how these tools might improve their AIs. There are a number of interesting choices available at present, and most have undergone one or more upgrades over the last year to become more robust and more capable:

- Biographic Technologies has upgraded its Maya AI plug-in and renamed it AI Implant. The tool is available both as a plug-in and as a stand-alone library for use by game developers.
- LouderThanABomb's Spark! fuzzy logic editor has gone the open-source route and is now available as a download, together with the Free Fuzzy Logic Library.
- MASA's DirectIA is still around and has been used in a couple of games, mostly of European origin. Version 3.0 is nearing release and will be available for Windows, PS2, and Xbox development.
- Mindlathe introduced a new AI middleware product called Pensor at GDC 2002 that offers some interesting capabilities.

It was clear from the conference and talking to various developers that there is a great deal of interest in the subject of AI middleware in general. Developers were pleased to see that so many new capabilities have been added to previous products and are generally willing to give them another look. Nobody I spoke to wanted to go on record that they were actually using AI middleware for their next projects, but several planned on asking for evaluation copies and taking a closer look.

One roundtable discussed the fact that it's possible that AI middleware won't gain widespread acceptance until some common specifications for game AI (possibly a standardized interface) evolves. The growing acceptance of other kinds of middleware products for graphics and physics may help get developers and producers used to the idea more quickly.

## Focus on the Future

At one of the AI roundtables the question arose, "Where will game AI be in five years?" There were lots of opinions on this, as one might expect, but there were some common themes among developers' discussions.

Developers felt that games in general will likely feature less scripting and more sophisticated interactive decisions on the part of the game's agents. AIs will make heavier use of layering and will perform more strategic planning with less emphasis on templates. We should also see better tactical coordination among AIs and their individual units than we're seeing now. There will continue to be expanded support for players to build their own AIs over time, if only because the tools that will enable this will be necessary for the designers and thus easy to release for general use with the game.

There will probably continue to be very slow investigation and adoption of the more exotic AI technologies such as neural networks and genetic algorithms. Even though a few games such as CREATURES and BLACK & WHITE make use of them, they're considered isolated cases.

AI middleware will continue to gain acceptance, and there will almost certainly be some breakthrough product or products that will make it more popular. The use of these products should eventually give rise to some common specifications for game AI that transcends game boundaries and makes possible the creation of dedicated AI hardware. There's a very long way to go to get there, though; at one roundtable we tried for half an hour to agree on a common definition of "a path" and couldn't even manage that.

We may also see more text-to-speech and speech recognition built into game AIs in the future as developers seek to make the experience more realistic. This still sucks up a lot of CPU cycles, however, so progress and demand will be slow. Developers are also considering the possibilities of supporting multiple CPUs now that we have widespread operating systems that can do that easily and how that might affect their designs.

It's amazing to see how game AI has evolved over time. It clearly continues to be one of the most innovative segments of the game industry and a huge driver in game design. Developers know what works and are making it work better. Advances that help are being incorporated slowly but efficiently. Reference works are coming out in increasing numbers that are helping developers share ideas, and AI middleware is finally becoming robust enough to be taken seriously. It all bodes well for the game AI developer in the coming years.

### FOR MORE INFORMATION

#### BOOKS

*Game Programming Gems* series, Vols. 1 and 2., Charles River Media, (Vol. 3 is due in July, 2002)

*AI Game Programming Wisdom*, Charles River Media, 2002 (reviewed on p. 10)

#### WEB SITES

AIWisdom.com
Gamasutra.com
Gameai.com
GameDev.net
www.gdconf.com/archives/proceedings/2002/homepage.htm

#### TOOLS

Biographic Technologies' AI Implant
www.ai-implant.com
LouderThanABomb's Spark!
www.louderthanabomb.com
MASA's DirectIA
www.animaths.com
Mindlathe's Pensor
www.pensor.net

jennifer olsen

# Game Developer's 2nd Annual Salary Survey

The past year has brought a lot of changes to the game industry, the job market, and the economy as a whole. The success of new and existing hardware last year helped pump an unprecedented $9.4 billion in total game-related sales into a sagging U.S. economy and has generated a lot of mainstream interest in game development careers.

While some outsiders have yet come to grips with the fact that there's more to making games than playing them all day, and some battle-worn industry veterans have absconded to higher-paying tech sectors (or any job with what resembles regular work hours), this survey represents the tens of thousands of U.S. professionals who make their living developing games.

This year's survey was conducted by research firm Audience Insights. In March 2002, 1,178 Game Developers Conference attendees took our comprehensive annual survey, of which the salary survey is one module, using on-site tablet computers. Then, in April, we e-mailed invitations to all *Game Developer* magazine subscribers and Gamasutra.com members asking them to participate in the survey and received 5,256 responses.

The survey data presented here is based on a total of 2,524 responses that remained after we eliminated responses that provided no numerical compensation data and those whose compensation figures were less than $10,000 or greater than $300,000 per year. We also eliminated responses that lacked certain demographic and classification information.

The sample represented in the salary survey data can be projected to the game development industry as a whole with a margin of error of 1.93 percent at the 95 percent confidence level. That means we can say with 95 percent certainty that the aggregate statistics reported in this survey would stay consistent within the margin of error across the entire population.

While the industry job market has remained healthy overall, it hasn't been immune to layoffs and other corporate casualties of an increasingly competitive marketplace. The past year's unprecedented success was wrought in no small part by the dogged work and incalculable overtime on the part of thousands of game developers. Game developers are known to thrive on challenge, though, and they got it in spades in the form of new hardware, changing market demographics, and relentless jockeying for position from publishers and hardware vendors.

What's the payoff for facing all these challenges? The overly simple answer appears on the following pages. But when a developer stands in a store and sees a game buyer longingly caress his or her creation, all that matters is the love of the game.

## PROGRAMMING

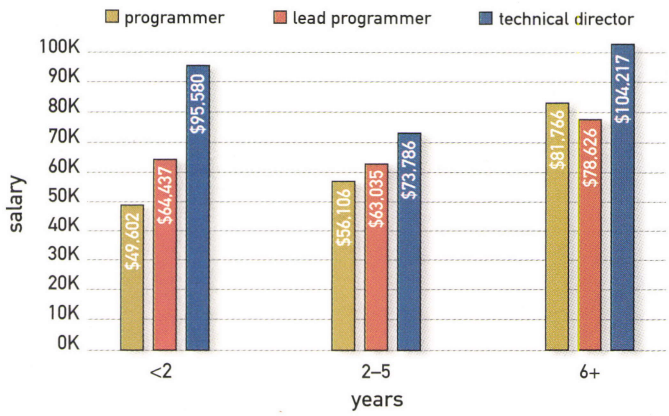
**Programmers always seem to be in demand, and accordingly** many find their jobs very demanding. The hours are long, the crunch modes interminable, the bug lists endless. Among the rank and file of programmers and senior programmers at most game companies you will likely find developers with a mastery of at least several of the industry's most in-demand skills: AI, networking, tools development, 3D math, physics, and preferably the ability to invert matrices in one's sleep.

With the growing focus on console game development, gameplay programming skills are rising in demand. Coding that may have been a virtual afterthought on a PC title now requires far more man-hours for fine-tuning the responsive kinds of gameplay favored by console game players. You can program in all the fluid dynamics simulations and volumetric fog you want in a scene pushing hundreds of thousands of polygons, but if someone holding the joypad isn't having fun with the controls, your sales (and perhaps your royalties) will suffer.

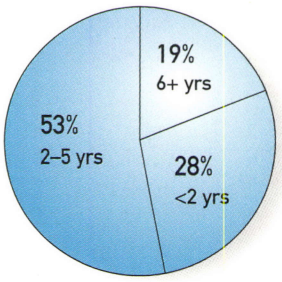
Experience and reliability are other sought-after qualities, ones which pay off in higher salaries for seasoned programmers. For those who have stuck it out for several years and have a proven track record, compensation increases accordingly to reflect both the employee's experience and the reduced investment risk on the part of the employer.

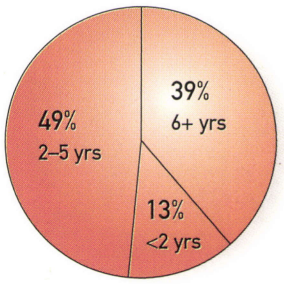
Programmers generally report to a lead programmer responsible for planning and scheduling programming tasks for a project. At companies with multiple projects, several leads may report into a technical director, who oversees programming productivity for the whole company and perhaps spearheads tools and technology development to be shared across teams. At single-project companies, these responsibilities often fall with those of a lead programmer upon a single individual.

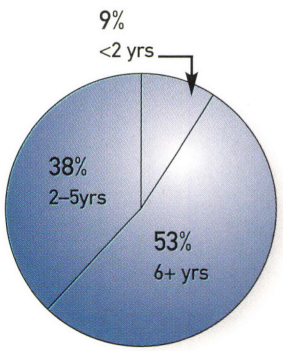
**Programming salaries per years of experience and position**



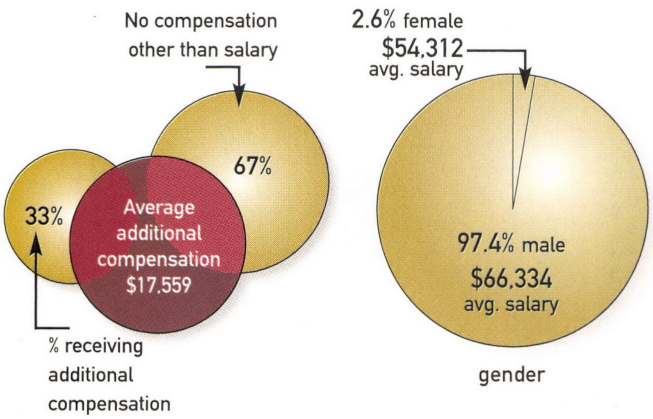

**Years experience in the industry**



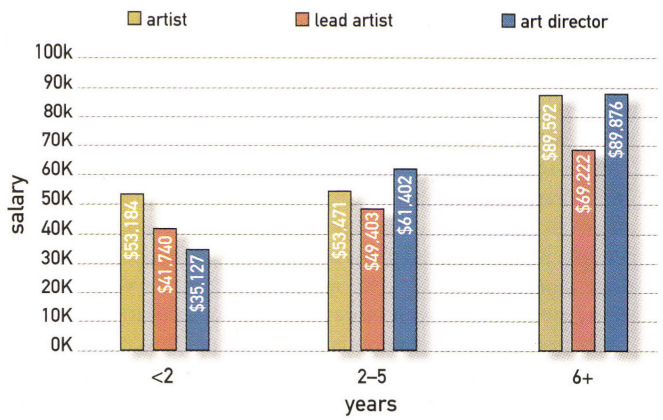

**All programmers**



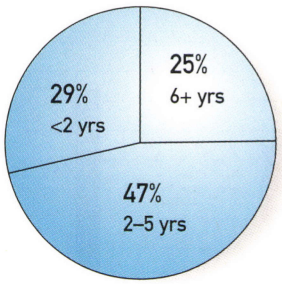

**Highest salary**
$300,000

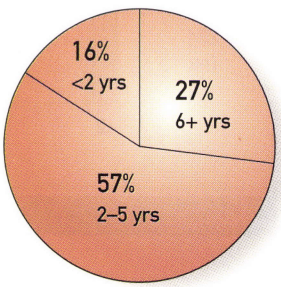
Note: Some percentages do not total 100 due to rounding.

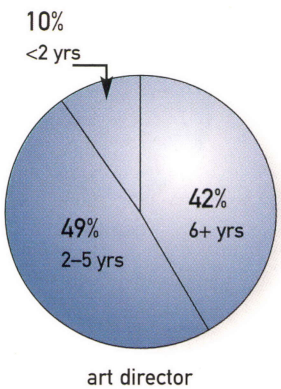
## Art salaries per years of experience and position




## Years experience in the industry



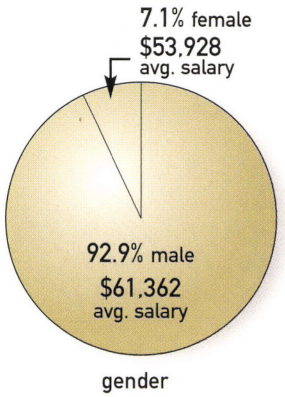

# ART

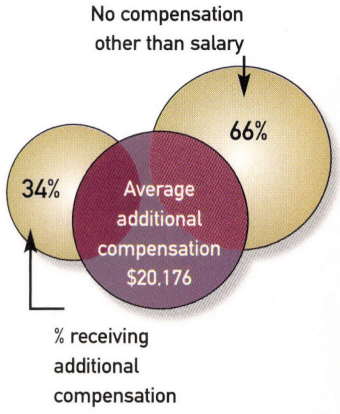
**Artists make up an increasingly significant chunk of game** development talent, as every generation of technology brings with it more polygons to be modeled, more characters to be animated, and more faces to be plastered with detailed textures. For the purposes of our survey, we considered as artists those who described themselves as artists, modelers, animators, texture artists, concept artists, and graphic or interface designers. We grouped lead artists and lead animators under the single classification of lead artist, those who manage and schedule teams of artists. At multiple-project companies, several leads might report into an art director, who might be responsible for making technology decisions and perhaps coordinating a certain look and feel across a range of products.

As with the other disciplines featured in this survey, experience pays. Hiring rookie artists unfamiliar with the rigid technical boundaries of game production environments can be risky when output demands are high and headcount is not. Clearly, though, there are rewards to sticking it out for a few projects, as compensation increases to where the most experienced artists can command much higher salaries.

The pay disparity between programmers and artists in the game industry is not a well-kept secret, but supply and demand is ever at play in any market, including the job market. Turnover and layoffs can be more tumultuous for artists as well, with the ebb and flow of art needs between major projects. However, one bright spot in our survey shows that roughly the same percentage of artists as programmers are being offered compensation plans above their base salary, and artists are taking home slightly more above-base compensation on average than programmers, which can help offset their generally lower base salaries.

## All artists




**Highest salary**
**$285,000**

# Current Hiring Trends

**How healthy is the job market in the game industry?**

Job seekers, employees, and industry observers have asked this question repeatedly over the last year. The answer is heartening: it's surprisingly healthy compared to what many people believe. It seems the recent economic downturn has not affected the game industry as dramatically as we have seen in other industries, though hiring practices have shifted somewhat. The two most important factors influencing hiring practices in our industry have been the platform transitions of the last two years and the increasingly hit-driven nature of the industry. Both have forced companies to think seriously about managing their payroll and hiring costs and maximizing their return on their investment.

We spoke with hiring professionals at game studios across the country to understand the changes in hiring practices that they have observed. While all agree that there have been differences, most companies told us they have not made significant changes in their approach to hiring. As the industry continues to thrive, hiring for specialized talent and experience is still difficult.

**What changes have occurred?**

Half of the HR professionals we talked to report that their company has forecast fewer openings than in previous years. The other half has experienced no change at all.

In general, job descriptions have evolved from a wish list to a longer list of detailed requirements. For example, a job listing which would formerly have read "Playstation 2 experience preferred" now reads "Applicant must have shipped at least one Playstation 2 title." This indicates not only increased selectivity on the part of the hiring authority but also the maturity of the platform in question. Also, as a result, entry-level applicants and recent graduates are capturing fewer of the available jobs.

**How has the hiring package been affected?**

While salaries still must remain competitive to secure good talent, some companies told us they no longer feel pressure to pay above market rate to secure the right candidate. Most said they have not made changes to their hiring bonus structure or other aspects of potential incentive packages. Some studios have reported using fewer signing bonuses.

International candidates who require visa sponsorship are having a more difficult time obtaining jobs in the U.S. Even when the economy was in full swing and "warm bodies with game experience" were at a premium, many companies in the industry did not want to consider visa sponsorship. There is even less inclination to consider sponsorship now. The extra costs in legal

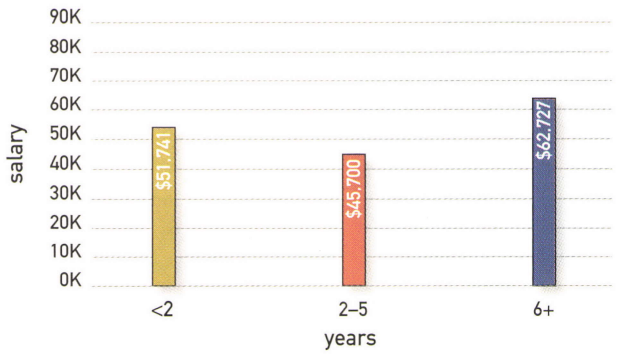
## DESIGN

**Game design is the development discipline that perhaps holds** the most cachet among lay folk, and not coincidentally it is both very competitive to break into and very hard to delineate in terms of required skills. Our survey considered designers to be those respondents who described themselves as game designers, level designers, lead designers, creative directors, and writers.

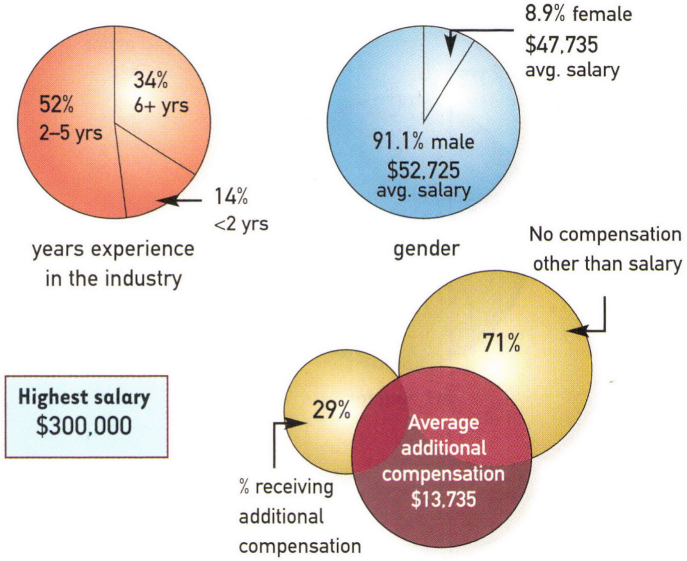
With so many people dreaming of breaking into a game design career, it's no surprise that salaries are low relative to other disciplines. But since having a good idea for a game and actually making a good game are two very different things, those designers who have the most experience, six years or more, are rewarded with more generous compensation.

Above-base compensation is also lower and less common among game designers compared to other disciplines, again likely due to the competitive nature of the job and relative paucity of positions available.

**Design salaries per years of experience**




**All design**



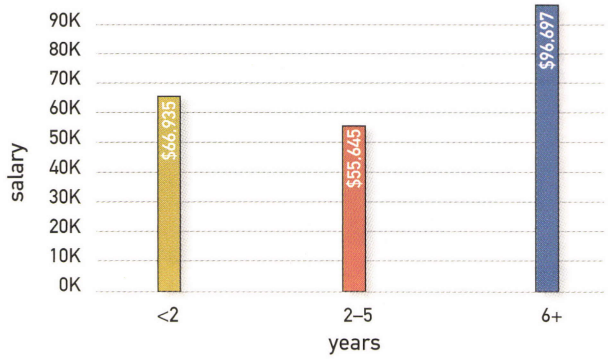

**Highest salary**
**$300,000**

## PRODUCTION

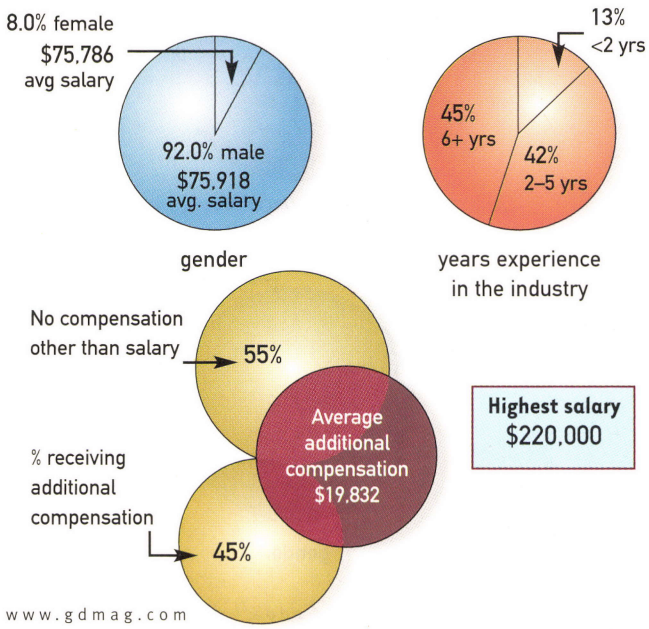
**A producer's job can run the gamut from devising budget plans** to managing QA to ordering takeout for the team during crunch modes. These are the people who simply do whatever needs to be done to get a game out the door on time, on budget, and of high quality. For the purposes of this survey, we considered a producer to be all those who described themselves as either a producer, associate producer, executive producer, or project lead/manager. Producers typically handle the day-to-day minutiae that spawn relentlessly throughout the course of a complex project and all its components. An executive producer might have additional roles within a company and thus oversee a game's development from a broader viewpoint, or may supervise several producers on several different teams.

Almost half the producers surveyed reported experience of six or more years, and their employers seem to like what they've done; their salaries far exceed their less experienced colleagues'.

**Production salaries per years of experience**




**All production**




fees, relocation expenses, and time lost waiting for the candidate to arrive in the country are not easily borne by project teams with time-critical deadlines.

**What resources are being used for hiring?**

The turnaround in spending, triggered by the dot-com bomb and 9/11, has brought thrift back in vogue, thus hiring costs are being scrutinized. Although the popularity of online boards has continued to rise, the number of candidates received from such sources creates a screening process nightmare.

Many companies are cutting hiring costs by decreasing internal staff and increasing outsourcing. Other companies are doing the opposite, cutting costs by decreasing outsourcing and increasing the use of internal staff for hiring. In large companies, outsourcing is generally used for high-level or specialized openings such as director of product development or VP of sales. Smaller companies continue to rely on outsourcing for most of their hiring needs. When managed effectively, both outsourcing and greater reliance on an already developed internal hiring staff continue to be useful cost-cutting tools to companies of all sizes.

**What is the reality?**

California is no longer the center of the game industry. As the game business grows, companies are spreading across the country. International companies are increasing in number as well. Hiring is becoming more of a business initiative, being forecast over the fiscal year. For the first time in our many years of recruiting, we had multiple incoming job requisitions this past December. This seems to indicate that the industry is focused more on the bottom line and won't stop thinking about their hiring needs due to year-end holidays.

One internal recruiter told us, "The feeling is that the market is flooded with qualified candidates, and therefore companies can be choosy. However, on the flip side, many candidates are not as desperate as companies would like to believe. Quality candidates will wait for the right opportunity at the right price."

**ANDREA COURTIE** | *Andrea has 12 years' experience in recruiting. Prior to joining Mary-Margaret.com, Andrea worked in-house or on exclusive contracts with many companies including Electronic Arts, Disney Online, Microsoft, Sierra Online, GT Interactive, OpenTV, and Sun Microsystems.*

**MARY MARGARET WALKER** | *Mary Margaret is one of the leading recruiters in the game business, having co-founded Mary-Margaret.com after six years of game development with Origin Systems and 3DO.*

## AUDIO

**The audio discipline includes those who describe themselves as** audio engineers, sound designers, composers, and the ever-popular "audio guy." With so many independent contractors plying their trade in the game audio business, this segment is perhaps subject to greater influence by the vagaries of the economy as a whole. Still, the most recent crop of game consoles has made sophisticated audio output a major selling point, and consumers seem responsive to the extra care developers are devoting to music and sound effects, a fact that should give designers and producers pause before simply farming out their game's audio to the lowest bidder.

Every generation removed from bleeps and blips that we are makes life better for game audio professionals. As tools improve and more development studios bring audio in-house with full-time audio personnel, the next few years should see an abundance of new and better opportunities for game audio professionals.

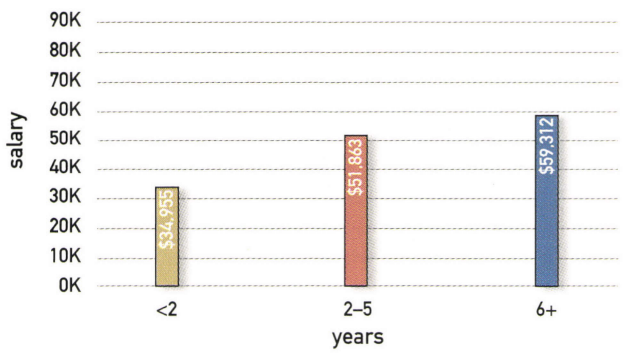
## OTHER TRENDS

**For developers feeling older and wiser than they were a year ago,** it's not their imagination. This year 31 percent of survey respondents reported being in the game industry at least six years, compared with just 19 percent in our 2001 survey.

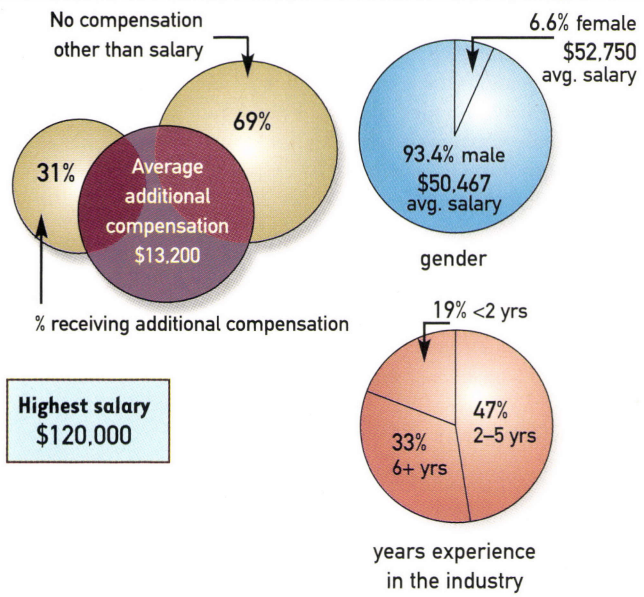
The western U.S. was both the best represented among our survey respondents and also the best paid. Clearly salaries rise with geographic competition where game development studios cluster, as can be seen with Texas far outpacing the rest of the southern states.

What can only charitably be called a gender imbalance still exists among game developers; this year's survey produced 5.2 percent female respondents overall, a slight drop from the 6.0 percent reported last year. Women in the industry on average made 89 cents on the dollar compared to men, which exceeds the national average of 76 cents on the dollar as reported by the Bureau of Labor Statistics for 2000, the most recent year for which such data is available.

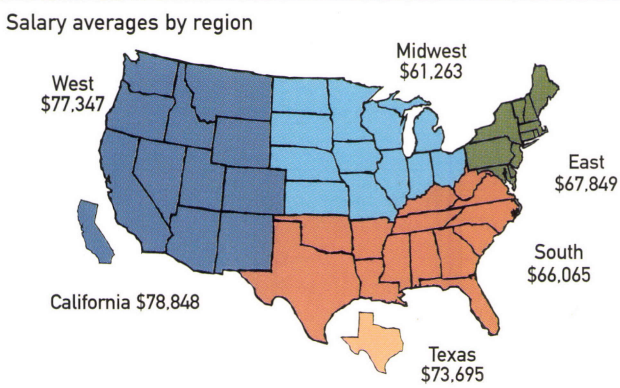
**Audio salaries per years of experience**

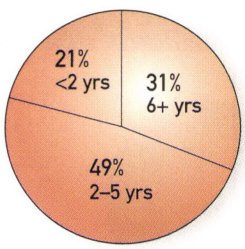


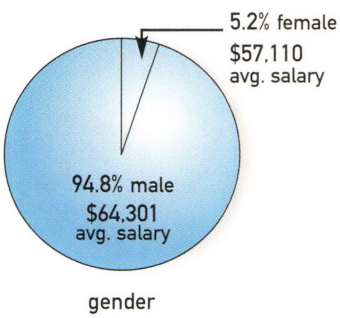

**All audio**




**Across all game industry**

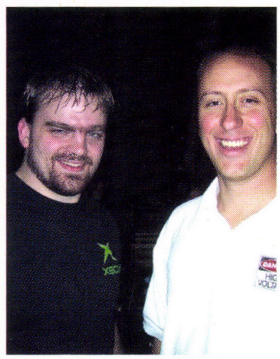


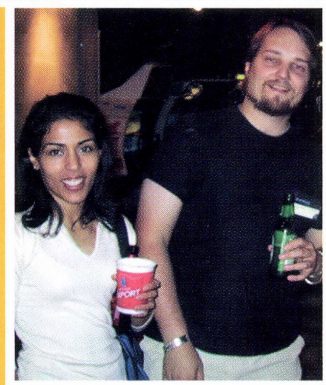



# UNITING

## the global game development community.





The IGDA is the independent, non-profit professional association for developers of interactive entertainment. Game developers have the potential to influence the business of games, inspire millions and establish games as the true art form of the next century. Without a collective voice and community, developers risk everything from lack of reward for their craft to the censorship of games – or worse.

Connect at local chapters like Los Angeles, Montreal, and London or get involved with the establishment of new chapters all over the world. Receive discounts on events, publications, products and services relevant to game developers.

**JOIN TODAY**

**www.igda.org**

igda international game developers association

The IGDA is sponsored in part by: Alias | wavefront®

# Surreal Software's DRAKAN: THE ANCIENTS' GATES

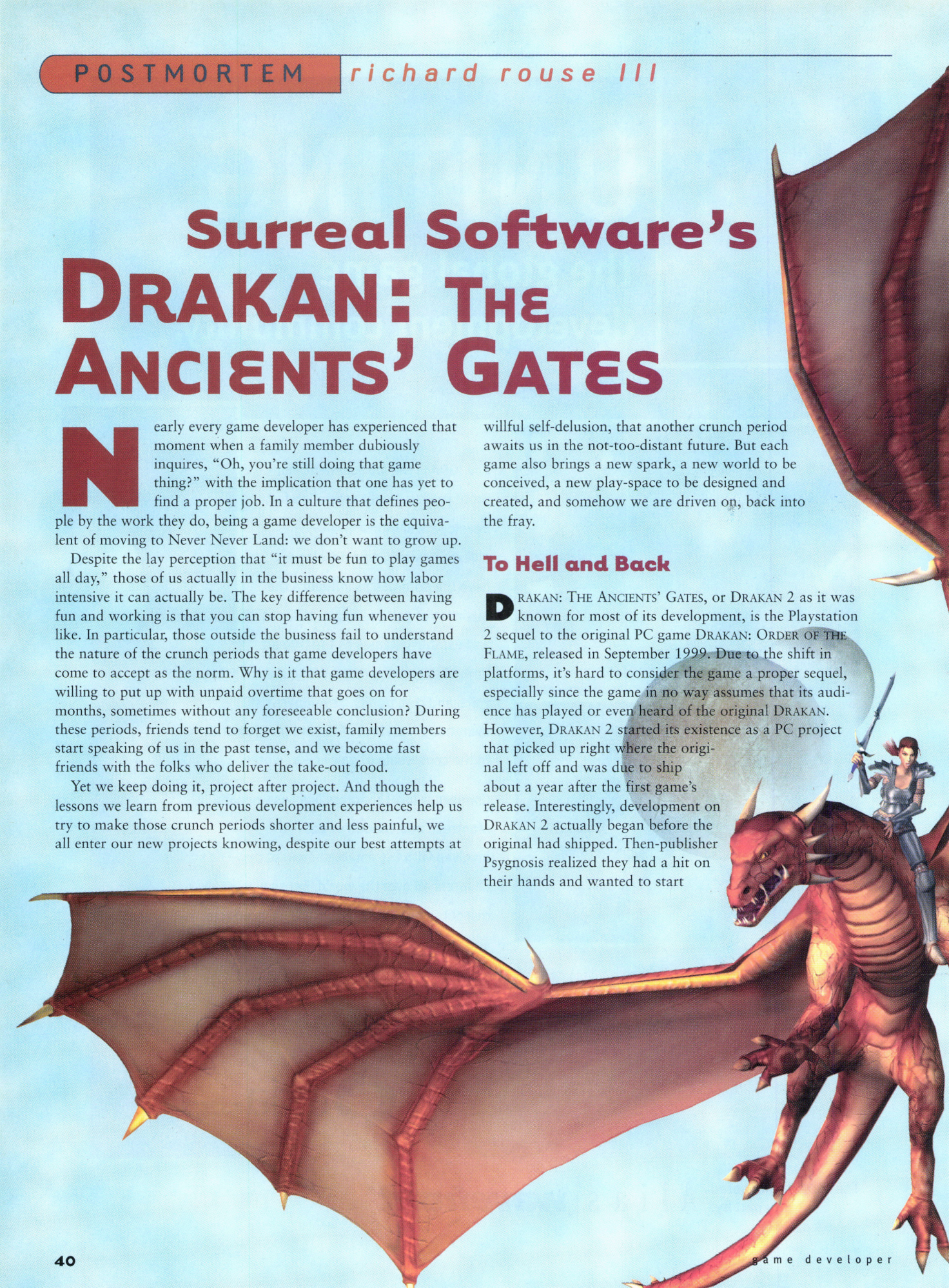
Nearly every game developer has experienced that moment when a family member dubiously inquires, "Oh, you're still doing that game thing?" with the implication that one has yet to find a proper job. In a culture that defines people by the work they do, being a game developer is the equivalent of moving to Never Never Land: we don't want to grow up.

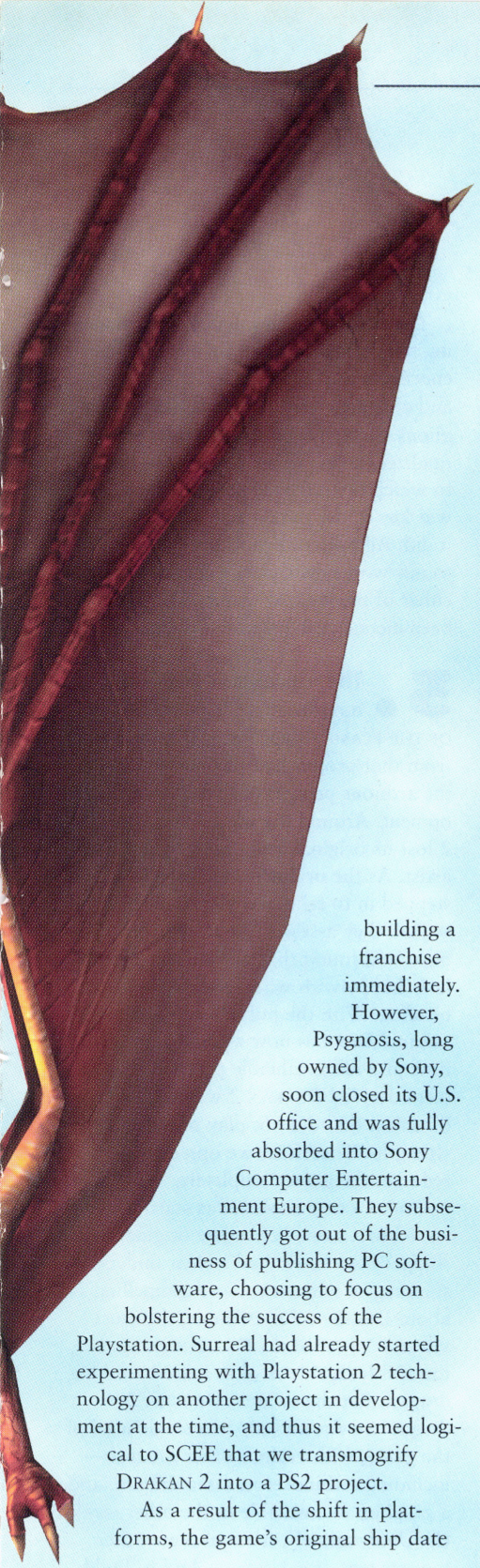
Despite the lay perception that "it must be fun to play games all day," those of us actually in the business know how labor intensive it can actually be. The key difference between having fun and working is that you can stop having fun whenever you like. In particular, those outside the business fail to understand the nature of the crunch periods that game developers have come to accept as the norm. Why is it that game developers are willing to put up with unpaid overtime that goes on for months, sometimes without any foreseeable conclusion? During these periods, friends tend to forget we exist, family members start speaking of us in the past tense, and we become fast friends with the folks who deliver the take-out food.

Yet we keep doing it, project after project. And though the lessons we learn from previous development experiences help us try to make those crunch periods shorter and less painful, we all enter our new projects knowing, despite our best attempts at willful self-delusion, that another crunch period awaits us in the not-too-distant future. But each game also brings a new spark, a new world to be conceived, a new play-space to be designed and created, and somehow we are driven on, back into the fray.

## To Hell and Back

DRAKAN: THE ANCIENTS' GATES, or DRAKAN 2 as it was known for most of its development, is the Playstation 2 sequel to the original PC game DRAKAN: ORDER OF THE FLAME, released in September 1999. Due to the shift in platforms, it's hard to consider the game a proper sequel, especially since the game in no way assumes that its audience has played or even heard of the original DRAKAN. However, DRAKAN 2 started its existence as a PC project that picked up right where the original left off and was due to ship about a year after the first game's release. Interestingly, development on DRAKAN 2 actually began before the original had shipped. Then-publisher Psygnosis realized they had a hit on their hands and wanted to start





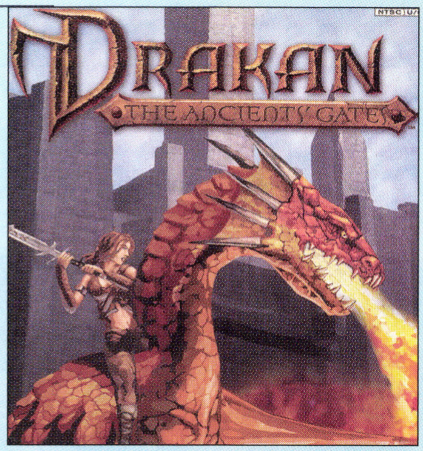
building a franchise immediately. However, Psygnosis, long owned by Sony, soon closed its U.S. office and was fully absorbed into Sony Computer Entertainment Europe. They subsequently got out of the business of publishing PC software, choosing to focus on bolstering the success of the Playstation. Surreal had already started experimenting with Playstation 2 technology on another project in development at the time, and thus it seemed logical to SCEE that we transmogrify DRAKAN 2 into a PS2 project.

As a result of the shift in platforms, the game's original ship date was pushed out a year, and Surreal adjusted to developing an entirely different type of game from what had originally been planned. Faced with the challenges of a new technology, drastically more complex art content, and a different style of gameplay, we set to work. The largest hurdle turned out to be technology, and the much-hyped power of the Playstation 2 did not arrive in any tangible form for a very long time. Also during this period, DRAKAN 2's publisher switched to Sony Computer Entertainment America, a publisher in our own time zone that was extremely understanding of the troubles we were facing and supported us through what was our darkest hour.

When the technology finally arrived, we were suddenly pushing more polygons than the more cynical among us had ever thought possible. There was a sea change in morale at Surreal, and we undertook a massive push to get two levels ready for the game's unveiling at E3 2001. This was by far the most satisfying part of the project for many team members, as the game finally became playable with a high polygon count and a solid frame rate.

With the technology finally established and our E3 demo a success, the project now needed to ship for the Christmas season. Although the time that preceded E3 had been intense for the team, the period that followed leading up to the game finally going gold in January 2002 was brutal, with people pushed to their breaking point. Amazingly, no one actually broke, and the game finally came together. But, as proud as the team is of the final incarnation of the game, some remain more proud of having survived the development that led up to it.

**RICHARD ROUSE III** | *Richard served as both a designer and programmer on DRAKAN: THE ANCIENTS' GATES. He is currently lead designer on an unannounced Playstation 2 title at Surreal Software. His previous games include CENTIPEDE 3D, DAMAGE INCORPORATED, and ODYSSEY: THE LEGEND OF NEMESIS. Rouse's somewhat hefty book about game design and development,* Game Design: Theory & Practice, *was published in 2001, with more information to be found at www.paranoidproductions.com. Feedback is encouraged at rr3@paranoidproductions.com.*



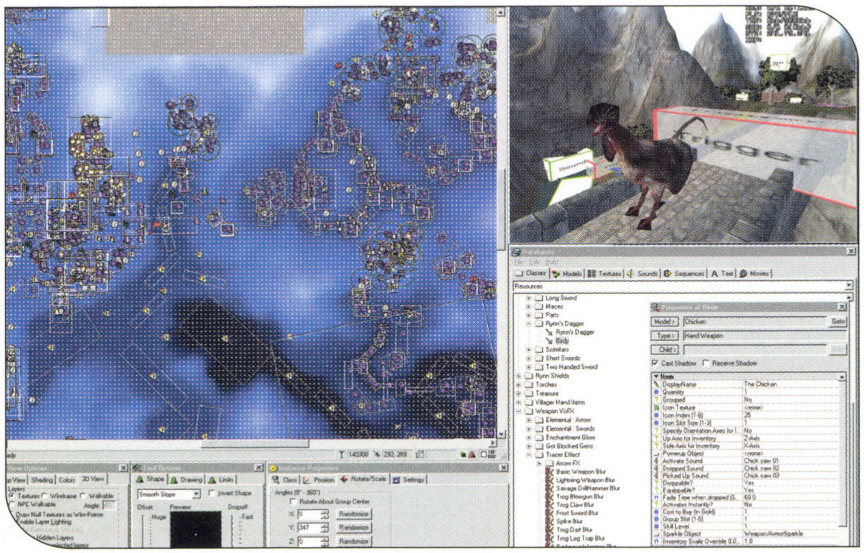

## GAME DATA

**PUBLISHER:** Sony Computer Entertainment America
**FULL-TIME DEVELOPERS:** 22
**PART-TIME DEVELOPERS:** 15
**CONTRACTORS:** 5
**LENGTH OF DEVELOPMENT:** Two and a half years
**RELEASE DATE:** January 2002
**PLATFORM:** Playstation 2
**OPERATING SYSTEMS USED:** Windows 95/98, Windows NT, Linux
**DEVELOPMENT SOFTWARE USED:** Microsoft Visual C++, ProDG, GNU C++, Visual SlickEdit, Photoshop, 3DS Max, Softimage, WaveLab, Pro Tools, Microsoft SourceSafe, SourceForge, SourceOffSite
**DEVELOPMENT HARDWARE USED:** Ranged from 600MHz to 1GHz Pentium IIIs and Athlons, typically with 256MB RAM and GeForce 2s.
**PROPRIETARY SOFTWARE TOOLS USED:** Riot Engine Level Editor, Riot Engine Modeler, Riot Engine Data Visualizer
**PROJECT SIZE:** 326,000 lines of C++ and assembly source code, 1,132 C++ classes, 1,152 source code files, 1,600 lines of dialogue, and approximately 450 CDs and DVDs burned.

## What Went Right

**1. Dedication of the team.** With the technology unfinished for a large period of the game's development and a resulting nine-month crunch time, DRAKAN: THE ANCIENTS' GATES could not possibly have been completed without a dedicated team that gave up their summer to see this game through to the end.

Despite doubling in size from the team that developed the original DRAKAN, Surreal remains a tight company of friendly people. Employees who have not gotten along with the rest of the compa-



The Riot Engine Level Editor provided the team with a lot of flexibility, streamlining the process of building the environments and setting up the gameplay.

ny have tended to depart quickly for one reason or another. There is a distinct lack of bureaucracy and office politics. Company outings occur regularly and generally relaxed policies allow for a remarkably enjoyable workplace. While definitely knowing how to have a good time, the team retains a high degree of professionalism and a strong work ethic. Through the stressful development cycle, there were surprisingly few blow-ups and a distinct absence of fistfights; everyone was too busy working.

So what made the team survive the crunch? First and foremost was a feeling of responsibility that everyone needed to pull together for the common good, so as not to let the other people in the company down. With a company culture such as Surreal's, the feeling of camaraderie is palpable. Second was a belief in what we were building. DRAKAN 2 was a high-profile project from a major publisher, and though there were moments of doubt, by and large the team felt that the game was destined to be great, if only we could manage to finish it. Given the number of problems that befell the game, DRAKAN 2 turned out better than it really should have because of the people who gave years of their lives to make it come together.

**2. Flexibility of the tools.** The entire DRAKAN 2 game world was rebuilt in nine months. Given the massive size of DRAKAN 2, this timeframe is staggering, and it was only possible because of the flexibility of our proprietary tools. DRAKAN 2 was built using a slightly upgraded version of the tools used for the original DRAKAN, and it is a testament to their continued flexibility that we were able to build a vastly more complex game using tools that are now several years old.

Our primary tool, the Riot Engine Level Editor, was built to be completely modular, allowing the game to grow to a massive size yet remain manageable. The editor allows for easy maintenance of a collection of databases which store models, animations, textures, sounds, scripted sequences, and movies. Designers are able to add and replace assets easily, which makes getting art into the game extremely simple. The 3D View window allows for the easy placement and movement of a camera in the game world, enabling designers to watch in real time as they make changes to the environment. With all of the rework that was necessary over the course of the project, our tool set's flexibility allowed us to adapt our content on the fly with a minimum of difficulty.

The tools were also perfect for balancing the gameplay, allowing designers enormous flexibility to tweak weapons and creatures quickly to suit specific situations. Since the game mechanics were finalized so late in the project, the degree to which the tools facilitated balancing was key to the game turning out as fun as it did. All told, our tools are such a boon to our workflow that the development of either of the DRAKAN games would have been inconceivable without them.

**3. The redesign.** Within a few months after DRAKAN: ORDER OF THE FLAME shipped in 1999, the team from that project had recuperated from the arduous process of that game's development. Around the same time, DRAKAN 2 lost its original lead designer and lead artist. As the original DRAKAN team stepped in to take over production of the sequel, they assessed the state of development and found that no one was particularly happy with where the project was heading. With the public's reaction to the original DRAKAN now available and with time providing valuable perspective, the team decided DRAKAN 2 was headed in a direction that did not play to the game's strengths. Therefore, we opted to redesign the game completely.

One of our mistakes was starting a sequel to a game before the original had shipped, before we had a clear understanding of what was most compelling about DRAKAN. For example, the most visceral aspect of the DRAKAN games is the ability to ride Arokh, a fire-breathing dragon. But the original DRAKAN 2 design had Arokh only in a single level of the game, with various other mounts — including a rhinoceros, a sea dragon, and a gryphon — available in the other levels. Unfortunately, none of these other mounts was as exciting as Arokh. Building on the strengths of the original game, DRAKAN 2's redesign made Arokh available in nearly every level.

This redesign resulted in throwing away a lot of work that had already been done, including art, levels, and partially implemented game mechanics. Though this early work had merit, it was not

appropriate for DRAKAN's sequel. The new game tried to use assets from the original design as much as possible, such as character types and locations, but not everything could be saved. Throwing away so much quality content was a bold move, but in hindsight the team members unanimously feel that it was the right decision and one that truly saved the project.

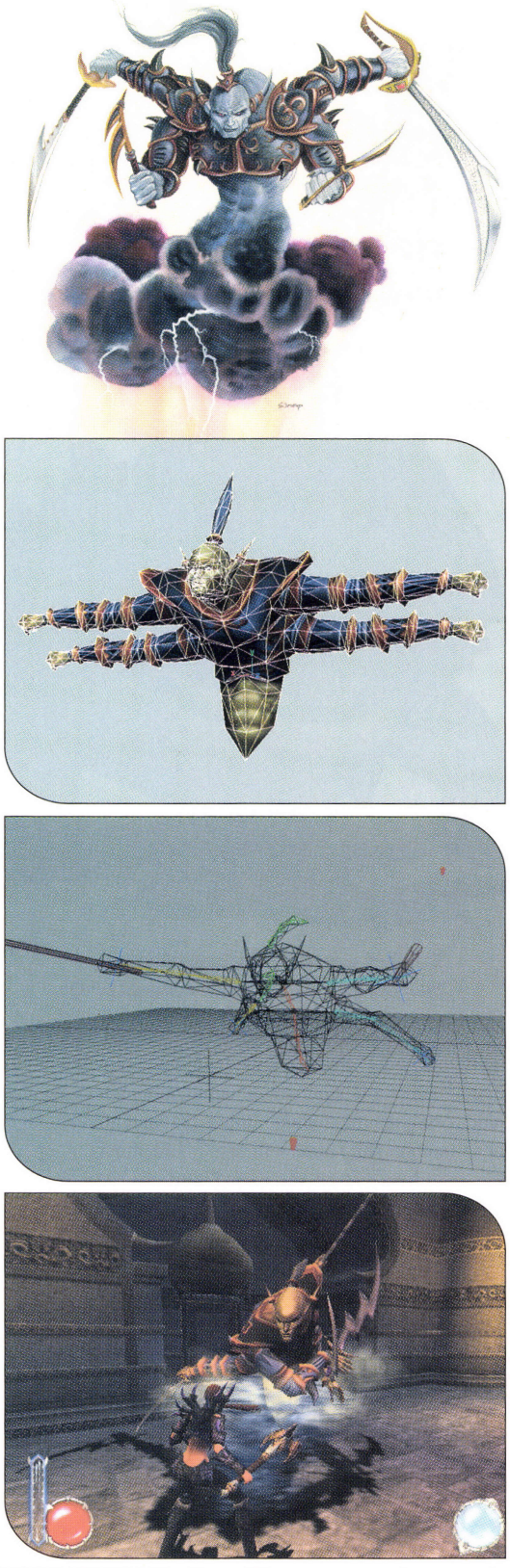
### 4. A sequel done right.

Many franchises have suffered when subsequent games in the series introduced little in the way of new technology or game mechanics. Players can end up feeling as if they've been cheated into purchasing the same game twice. DRAKAN: THE ANCIENTS' GATES improved over the original in the power of the engine it used, the gameplay mechanics it employed, and the overall presentation of the game.

From a graphical standpoint, DRAKAN 2 continued in the distinctive visual style of the original and upgraded it to take advantage of the Playstation 2. To look at the two games side by side, DRAKAN 2 obviously looks superior and more highly detailed, but at the same time the games are clearly of the same lineage and art style with their distinctive, hand-drawn look.

From a game-design standpoint, the game delivered the strongest features of the first game while adding new elements that deepened the player's experience. In particular, we made DRAKAN 2 less of a strict action experience and more of an action-RPG hybrid. Through adding player classes, magic, a skill system, and an economy, DRAKAN 2 became a richer experience for the player.

The team was also careful to listen to the feedback we got from the first game. One of the biggest complaints from the original was about the writing and story. This time we devoted a single in-house writer to the project and secured vastly superior voice acting, which benefited the storytelling aspect of the game. DRAKAN 2 ended up incorporating enough original content to give players a truly new experience instead of just a level pack.



TOP TO BOTTOM. The Storm Djinn enemy as color concept, model in 3DS Max, skeleton in Softimage, and final in-game character.

### 5. PS2 technology.

Though our PS2 technology was an extremely long time in coming, when it finally did arrive it blew us all away. For example, in the original DRAKAN we had a scene limit of 3,000 polygons. In DRAKAN 2, Rynn, the main character, is made up of that many polygons alone.

Our technology was composed of a number of core components, including a stripping system for efficient model rendering, an instance renderer for drawing a large number of identical models extremely cheaply, an efficient quaternion-based skeletal animation system, a "splat" texturing technology for procedural blending of landscape textures, and a dynamic data-loading system that allowed us to build massive levels while still fitting within memory.

Coupled with these features was a range of visualization and debugging tools that allowed developers to see easily what was taking up processing time in a given situation, so levels could be optimized for peak efficiency. Our Riot Engine Data Visualizer allowed us to track the use of each piece of memory, providing easy navigation of memory dumps to see what line of code was allocating what block of memory. By allowing us to track down memory leaks readily and identify what parts of the code were using up more memory than appropriate, this tool became crucial to squeezing our massive game into the PS2's 32MB of memory.

Most importantly, our PS2 technology was robust enough to be immediately put into use on our

new PS2 projects. Though we could have used it a lot sooner, its power made up for its late arrival.

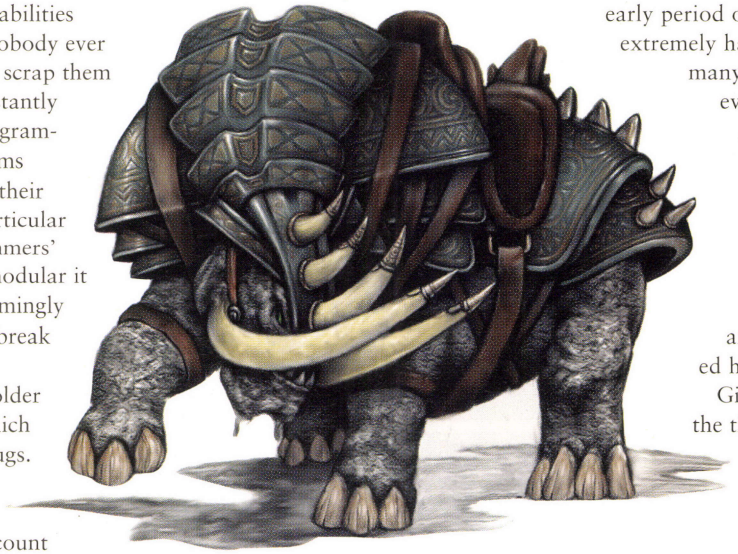
## What Went Wrong

**1. Bugs.** DRAKAN: THE ANCIENTS' GATES was built on the code of the original DRAKAN, code that is now upwards of six years old. Many of these older systems were known liabilities going into the project, but nobody ever thought that we had time to scrap them and start from scratch. Constantly moving deadlines forced programmers to bandage ailing systems rather than put them out of their misery. Our AI system in particular was the bane of the programmers' existence; instead of being modular it was monolithic, and any seemingly simple change was likely to break 20 other behaviors.

In addition, some of our older systems didn't scale well, which led to more unanticipated bugs. Effects, collision, and AI systems that worked fine in low-polygon, low-object-count environments were brought to their knees or broke completely when placed in the more complicated world of DRAKAN 2.

Our lack of any sort of testing early on in the project magnified the bug problem. Programmers implemented systems that would then not be used by the designers or even other programmers for months. Furthermore, not all of the bugs were even programming related: half of the game-breaking bugs in our database were design and art issues. Our tools put a lot of power in the designer's hands to change behaviors with a minimum of hassle and to create effects that programmers never anticipated. But with great power came great responsibility, and when the tools were not used carefully, their flexibility tended to create bugs.

For our post–DRAKAN 2 projects, we are starting out with a completely clean slate of gameplay code. Furthermore, we have implemented a strict code review system where multiple programmers go over all code before it is added to the project. We are now much more dedicated about testing new systems as soon as they are added to a build. As a result, the projects we now have in early development are much less buggy than DRAKAN 2 was when it entered beta.



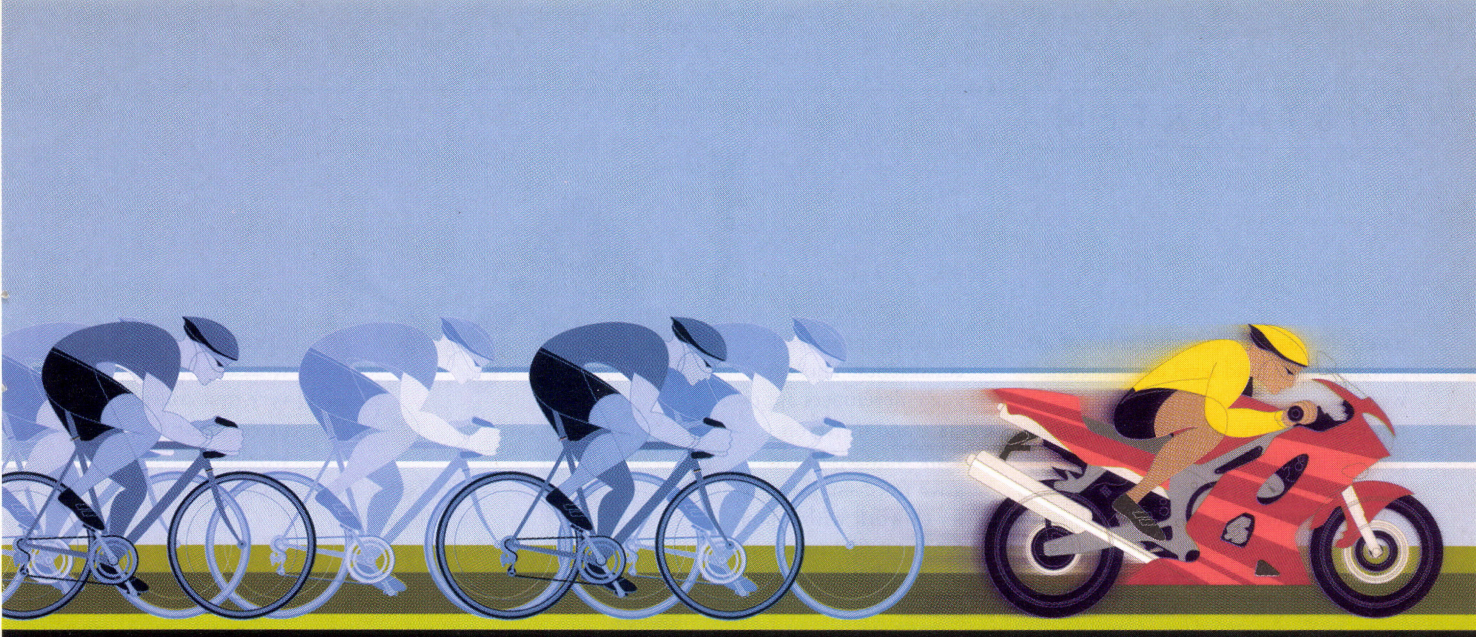
The rhino was one of the three alternate mounts Rynn could ride in the original design.

**2. Technology was a long time coming.** For almost a year, the art and design teams' mantra to the technology team was "When are you going to get the frame rate up?" The initial conversion of the Riot Engine to the PS2 had gone extremely smoothly. The original Drakan was up and running at 15fps within a few months, with all rendering occurring on the PS2's core processor. Initial estimates were that with another two months of work numerous systems could be moved to the VUs and we would have fully unlocked the promised power of the PS2. Since then, the PS2 has earned a reputation as a challenging system to develop for, but back when we started everyone had bought into the Sony hype machine and its startling technical demos.

Unfortunately, those two months we had planned on turned into a year. No one on the programming team anticipated just how difficult it would be to get a powerful engine running on the PS2; the PS2 simply does not have the linear development curve Surreal was accustomed to on the PC. The PS2's curve is more exponential, resulting in a long period of minor technological improvements before the engine suddenly becomes extremely powerful. During this early period of development, it's extremely hard to predict just how many polygons an engine will eventually be able to push. As expectations about what we would be able to do shifted, the art team first built too-high-polygon objects and then too-low. Once the technology was solidified, all of the art that had been constructed had to be rebuilt.

Given how much we knew at the time, there is not much we could have done to avoid our technology delays. We could have put all our programmers on the PS2 immediately and not required the new engine to be backwards compatible with our old content. But since we switched platforms mid-project, these were not really options. Fortunately, as I mentioned previously, our finished technology turned out to be quite powerful; the steep PS2 learning curve was simply something we had to overcome.

**3. Lack of planning.** We planned out DRAKAN 2 to a much greater extent than the original DRAKAN, and the early redesign correctly identified many problems and rectified them. But at the same time, the redesign did not go far enough to fully plan out the game. People implemented systems and built whole levels before the team realized that something was never going to work from a gameplay standpoint. For example, a massive underwater section of the game, in which Arokh was able to swim, was jettisoned when it turned out to be fundamentally flawed. Unfortu-



# FREE YOURSELF TO DO THE CODE THAT MATTERS

**@ 60 HZ...**

- 25 High-Resolution Blended Lit Textured Characters with Shadows, 525 Bones (21 per character, 4 weights per vertex)
- High-Resolution Static Textured Environment
- 3-D Overlay GUI, Stereo Sound, AI, Controllers, Particle Systems, Shadows, Shaders
- Additional Animated Skinned Crowd Characters (based on console)

**60 HZ**

- One Source
- One Database
- One Intrinsic Alchemy Platform

**PS2 Xbox* Gamecube** PC**

**Try it for yourself!**
**Download Free ArtistPacks for 3ds max™ and Maya™ at www.intrinsic.com**

*Xbox adds 95 extra characters with 760 bones
**Gamecube adds 12 extra characters with 96 bones

## INTRINSIC ALCHEMY™

THE GAME DEVELOPER'S ULTIMATE TOOL

sales@intrinsic.com

INTRINSIC GRAPHICS™

WWW.INTRINSIC.COM
650.966.6800

©2002, Intrinsic Graphics, Inc. All Rights Reserved. Intrinsic Alchemy and the Intrinsic Graphics logos are trademarks and/or registered trademarks of Intrinsic Graphics, Inc.

nately, it was only cut after a lot of art, programming, and level design work went into it.

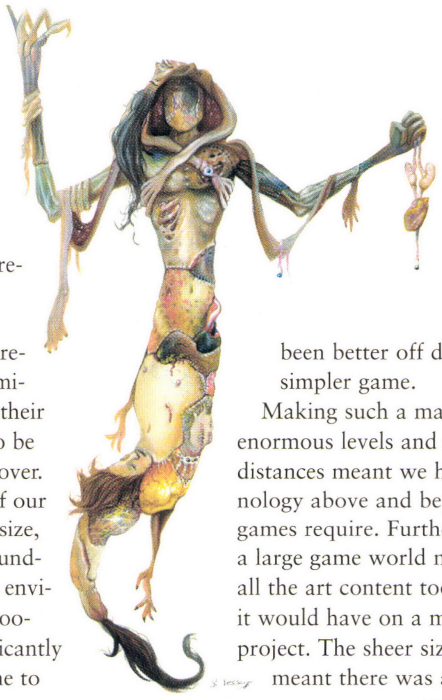
Certainly game design is an organic process, and no team is ever going to be able to plan out a game completely in a design document. Good ideas will come up that no one thought of before the game was actually running, while systems that seemed like good ideas in preproduction turn out to be tedious in practice. Nevertheless, some amount of planning, even if it is only 75 percent accurate, is better than a plan that only fully considers 25 percent of what will take place in the game. More improvisational game design can work with smaller teams, but with a team of 30-plus, it is invariably a scheduling nightmare.

Since DRAKAN 2, Surreal is using a much more intensive design process in preproduction. We fully map levels out on paper before finalizing the design, giving designers a chance to plan where various mechanics will be used and allowing artists to assess how difficult each environment will be to build. This process lets us understand better what will work in terms of both gameplay and scheduling, and we can make adjustments before any actual implementation has begun.

**4. Lack of approval process.** Over the course of DRAKAN 2's development, content was often created without anyone looking it over properly to see that it matched the game's overall vision. As a result, team members added art, code, or levels that remained in the game for months before someone would notice that they were inappropriate for one reason or another.

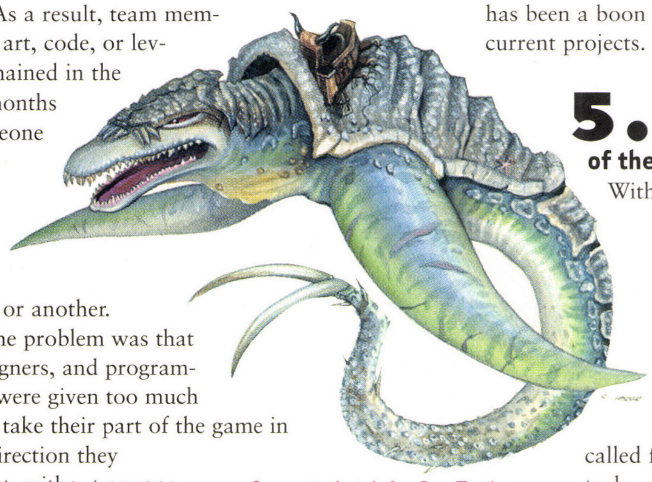
Part of the problem was that artists, designers, and programmers alike were given too much freedom to take their part of the game in whatever direction they thought best, without anyone checking for overall consistency. Though game developers appreciate and thrive on some amount of creative freedom, they prefer to work within limitations if that means their work will not need to be redone several times over.

With the growth of our team and the game's size, our problems compounded. In a smaller team environment with a not-too-large game, it's significantly less work for everyone to know what everyone else is doing and to keep an eye on all aspects of development. For DRAKAN 2, our team size doubled and the game size quadrupled without any change in the amount of oversight.



Early concept sketch of the Flesh Mage boss.

DRAKAN 2 matured us enough to realize that large-scale game development is not a purely creative process, but requires a strict production pipeline where problems can be discovered early and fixed quickly. Toward the end of DRAKAN 2's development, Surreal implemented a highly structured content review process with a number of different stages for any given game asset. This all but eliminated rework during the final months of DRAKAN 2's development and has been a boon to our current projects.



Concept sketch for Sea Turtle.

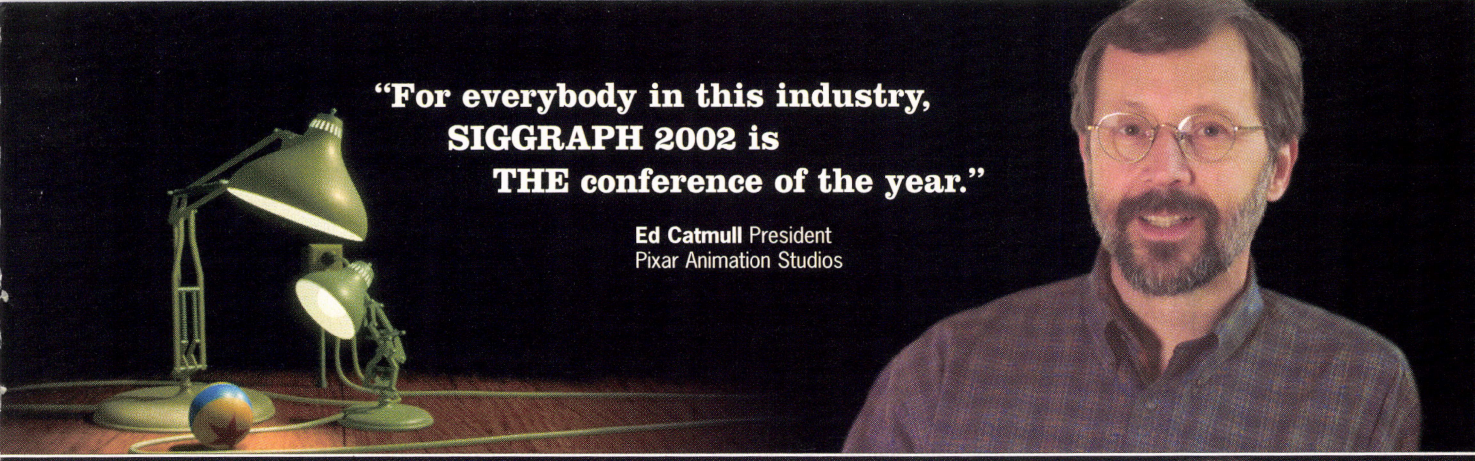
**5. The scope of the game.** Without a doubt, DRAKAN 2's fantasy setting and RPG trappings called for a truly epic experience. Yet many wonder if, for our very first console project, Surreal wouldn't have been better off developing a smaller, simpler game.

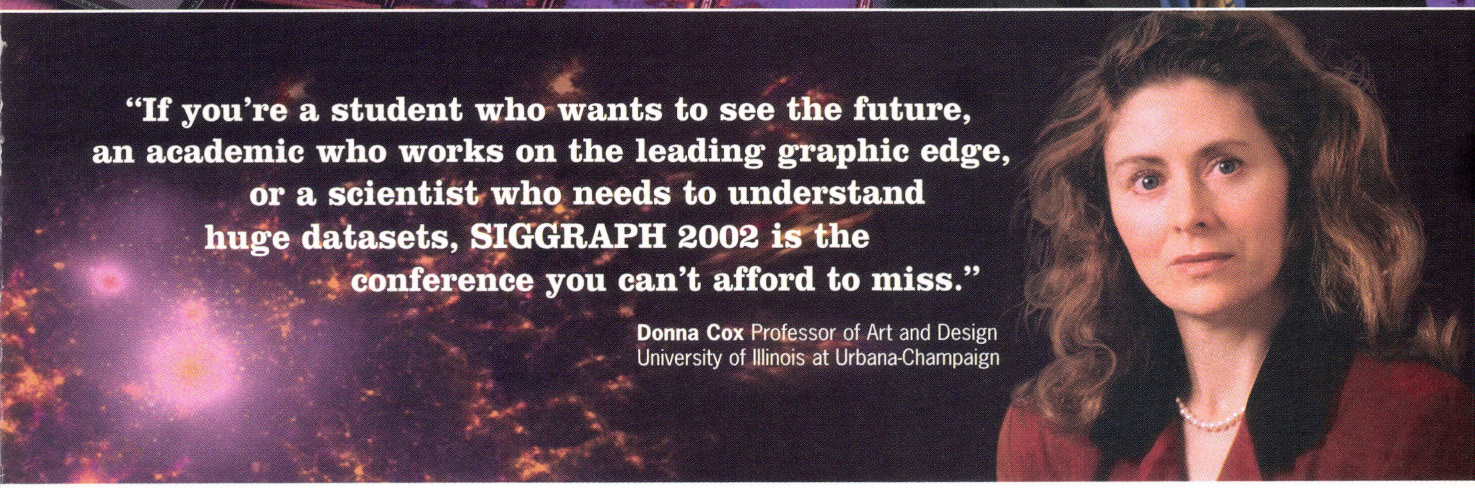
Making such a massive world with enormous levels and extremely far draw distances meant we had to push our technology above and beyond what most games require. Furthermore, having such a large game world meant that building all the art content took twice as long as it would have on a more reasonably sized project. The sheer size of the game also meant there was a similarly colossal amount of bugs, compounding our problems in that department. It's impossible for anyone on the team to look back now and not take some pride in how huge and epic the game is, yet taking into account a similarly epic crunch period, team members are forced to wonder what compelled us to make such a damnably big game.

## Last Call, Last Call

A person's time and labor are the most precious things he or she can give. As game developers, we give a lot of our time to the teams we work with, to the games we develop, and in turn to those who play the games we make. It's well nigh impossible to explain to people outside the game industry why exactly why we do it, yet many of us barely give it a second thought. What else would we do with ourselves?

There's something extremely exciting about creating something from nothing. All artists do this to some extent, but game developers are the only ones who create worlds that are built for others to explore and discover in their own way. For all the advantages that games have over media, there is an equal trade-off in terms of the enormous commitment that they require from their teams. For game developers, the payoff in the final product makes all the time invested worth it. Otherwise, we wouldn't still be here.






SAN ANTONIO

# SIGGRAPH 2002

**LEARN** the next generation of powerful hardware and software. **UNDERSTAND** how technical innovations are changing your work, your profession, your company. **APPLY** your new knowledge to creative and business breakthroughs. When you leave San Antonio, your future will be transformed.

**www.siggraph.org/s2002**

Conference: 21-26 July | Exhibition: 23-25 July | Henry B. Gonzalez Convention Center | San Antonio, Texas, USA

NEW RESEARCH FROM THE GAMA NETWORK

# Getting in the Wireless Game

## Opportunities and Entry Points for Game Developers

With projections of mobile gaming reaching global revenues of $5 billion by 2005, you can't afford to ignore this emerging market.

Get accurate, in-depth analysis of this market's technologies and momentum in this comprehensive research report:

- ✔ Identifies key players in the mobile market
- ✔ Explains the varied paths of entry for game developers
- ✔ Includes an overview of wireless platforms and compares capabilities
- ✔ Helps PC, console and web-based developers find the best way to take content mobile
- ✔ Explains the business and partnership opportunities between content creators and media companies
- ✔ Examines emerging opportunities for game developers

**Purchase this report at**
**www.gamasutra.com/store**
**using code GDMAGWIRELESS02 to**

**SAVE $400**

**Regular price: $795** ***Your price: $395***

mbusiness RESEARCH
INSIGHT ACTION ADVANTAGE

GamaNetwork

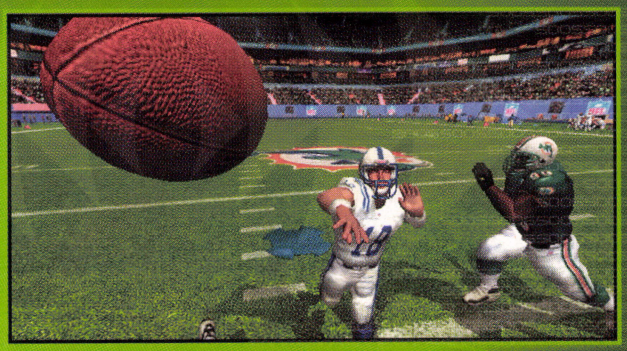
# ARE YOU EXPERIENCED?

***Software Engineer - Animation***
We're looking for an experienced game developer who will help craft new animation and physics components for upcoming titles on the Xbox. A strong background in 3D physics, kinematics, and mathematics is required. Detailed understanding of 3D hierarchical rigid-body animations, motion blending, physics-based collision detection/response, and forward/inverse kinematics is preferred.

***Senior Software Engineer - Graphics***
We have an exciting opportunity for a senior graphics developer to join us in pushing the envelope of the Racing and Driving genres for Xbox gaming. Primary responsibilities include designing, implementing, and supporting the graphics pipeline of a key internal title. Candidates should have extensive knowledge on shadowing techniques, lighting and multi-texturing, effects, visibility algorithms, particle systems, and, in general, could be considered an expert in the field. A B.S. or M.S. in Computer Science or a related field and at least three years of relevant experience are required.

***Senior Software Engineers - Graphics***
We're looking for several talented senior graphics developers for Xbox sports titles. The team will push the Xbox to its technical limits; graphically this means complex lighting effects, bump-mapping, motion blur, and other pixel and vertex shading techniques. Recent Direct3D expertise is required.

***Senior Software Engineer - AI***
AI - Just two vowels found in the English language or something much more compelling? We're looking for a developer with a passion to create a cutting-edge AI system for an Xbox Hockey game. This system will not merely react to the game player's input, it will anticipate and adapt. Position Requirements: Degree in Computer Science or Computer Engineering with a focus on AI; Strong C++ skills; Strong communication skills.



***Software Engineer - Physics***
We have an exciting opportunity for a physics developer to join us in pushing the envelope of the Racing and Driving genres for Xbox. The primary responsibilities include owning and implementing the vehicle dynamics, collision detection, and rigid body physics. Candidates must have a thorough understanding of these areas and how they apply in a game environment. A college degree in a related field and at least three years of relevant experience are required.

***Game Designer***
Envision, document, and champion the play and functionality of a game from start to finish including the story, core feature set, game shell, animations, and game flow. Candidates must be able to communicate clearly and concisely with various members of the team via written and verbal means. A minimum of two years experience in a design role at a game company or four years experience in a related industry is required.

Please visit http://www.microsoft.com/careers.htm to see the full list of available positions.

When submitting your resume to gamejobs@microsoft.com please include on the subject line the job code Aq24u0702. No game concept submissions please.

**Microsoft** game studios

© 2002 Microsoft Corporation. All rights reserved.
Microsoft is a registered trademark of Microsoft Corporation.
Microsoft is an equal opportunity employer and supports workplace diversity.

XBOX

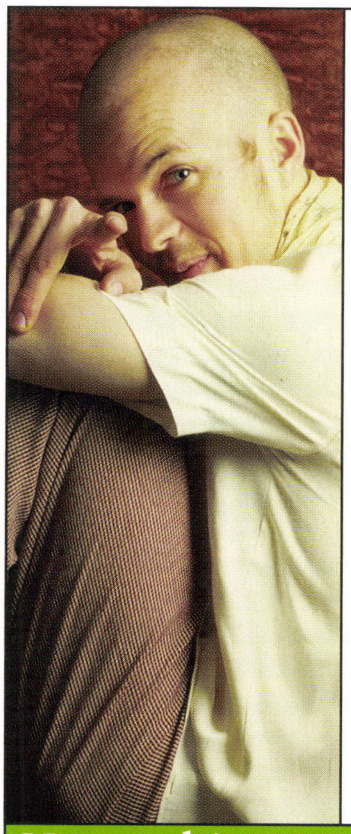
Konami offers a comprehensive benefits package and is searching for driven individuals with a passion for games.

We currently have positions open for:

*Lead Programmer
*Programmer
*Artists
*Entry-Level Artists
*Sound Designer

Please contact us at KCEAHIHR@konami.com or visit our website at:
www.konamihwi.com/jobs.php

KONAMI



Gamasutra.com

It's simple:

1. Search thousands of resumes by profession, game credits, technical skills, or preferred employment status.

2. Find experienced, professional game developers with **an average of 5 years of industry experience** and **more than 4 titles developed.**

3. Perform unlimited searches during your access period.

GamaNetwork

How to hire a game developer.
www.gamasutra.com/resumes

Recruiters With A Different Approach!

**Studio Search, Inc.** is the leading recruiting firm for the Interactive Industry. Founded in 1993, Studio Search, Inc. specializes in recruiting highly qualified Programmers, Artists, Producers, Developers, Sales and Marketing Professionals.

Our relationship with each candidate is based on trust and respect. All fees are paid by our client-companies. All resumes are treated discreetly and confidentially.

What sets us above the others is:

- Respect for your valuable time by only submitting your resume to our client-companies with open job postings.
- Obtaining your permission prior to forwarding detailed information and resume to our client companies.
- Our ability to confidentially evaluate your resume and match your requirements with open job postings.
- We take a genuine interest in your personal and professional goals and appropriately match your skills and location adaptability.

We encourage you to submit a current resume via E-mail to:
info@studio-search.com

**www.studio-search.com**
**1-800-822-4102**

STUDIO Search

**Remember when games were fun? They still are at Blue Shift!**

**If great games are in your future you belong at Blue Shift.**

We have immediate openings for programmers, artists & animators. Blue Shift is located in the heart of Silicon Valley. Close to Stanford University. Steps from CalTrain, fabulous restaurants, cafes, and more...

blue shift

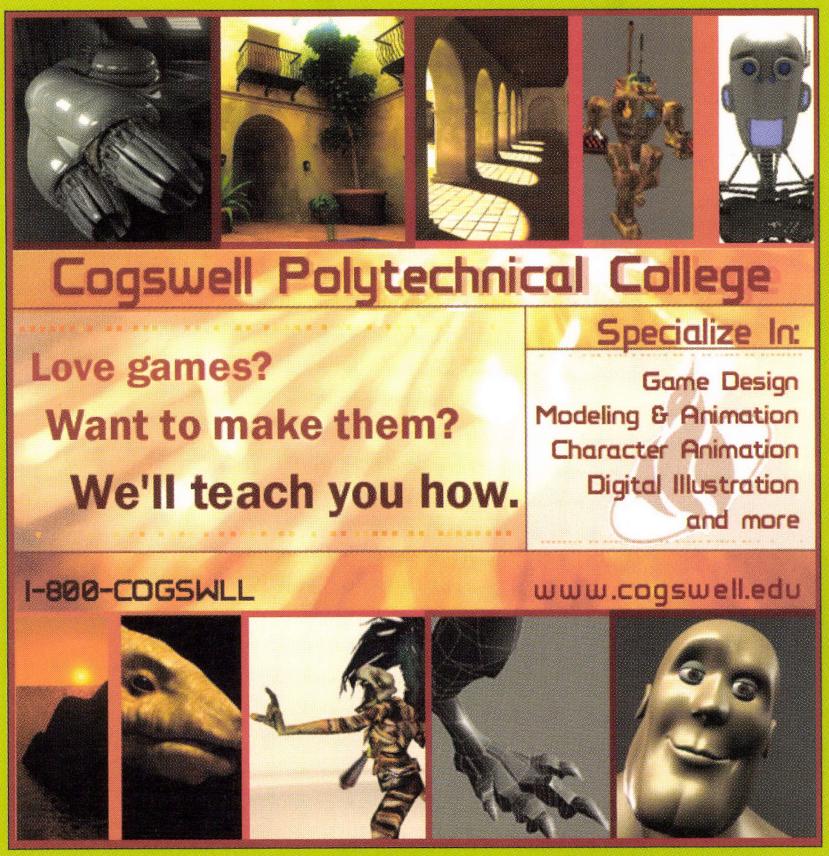
**Come join us. And have some fun!**

Email your resume to:

**Jobs@blueshiftinc.com**

Or FAX it to us at:
650.614.4856

**200 S. California Avenue, Suite 201 Palo Alto, CA 94306 650.614.4855**
**www.blueshiftgames.com**

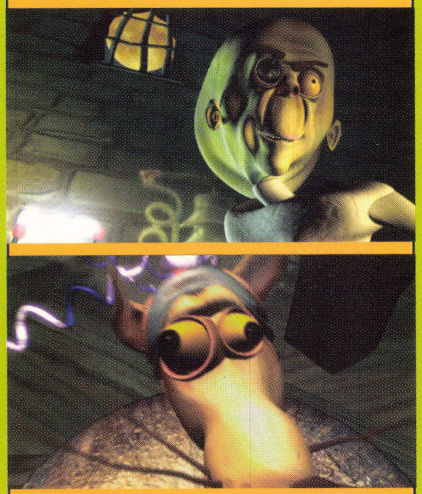


If you squeeze the new DVD from VFS you will never again be confused about who's got the juice.

To get more information or your own DVD of VFS student work call 1-800-661-4101 or email dvd@vfs.com

VFS **Vancouver Film School**
Creative. Disciplined. Focused.

Savannah College of Art and Design

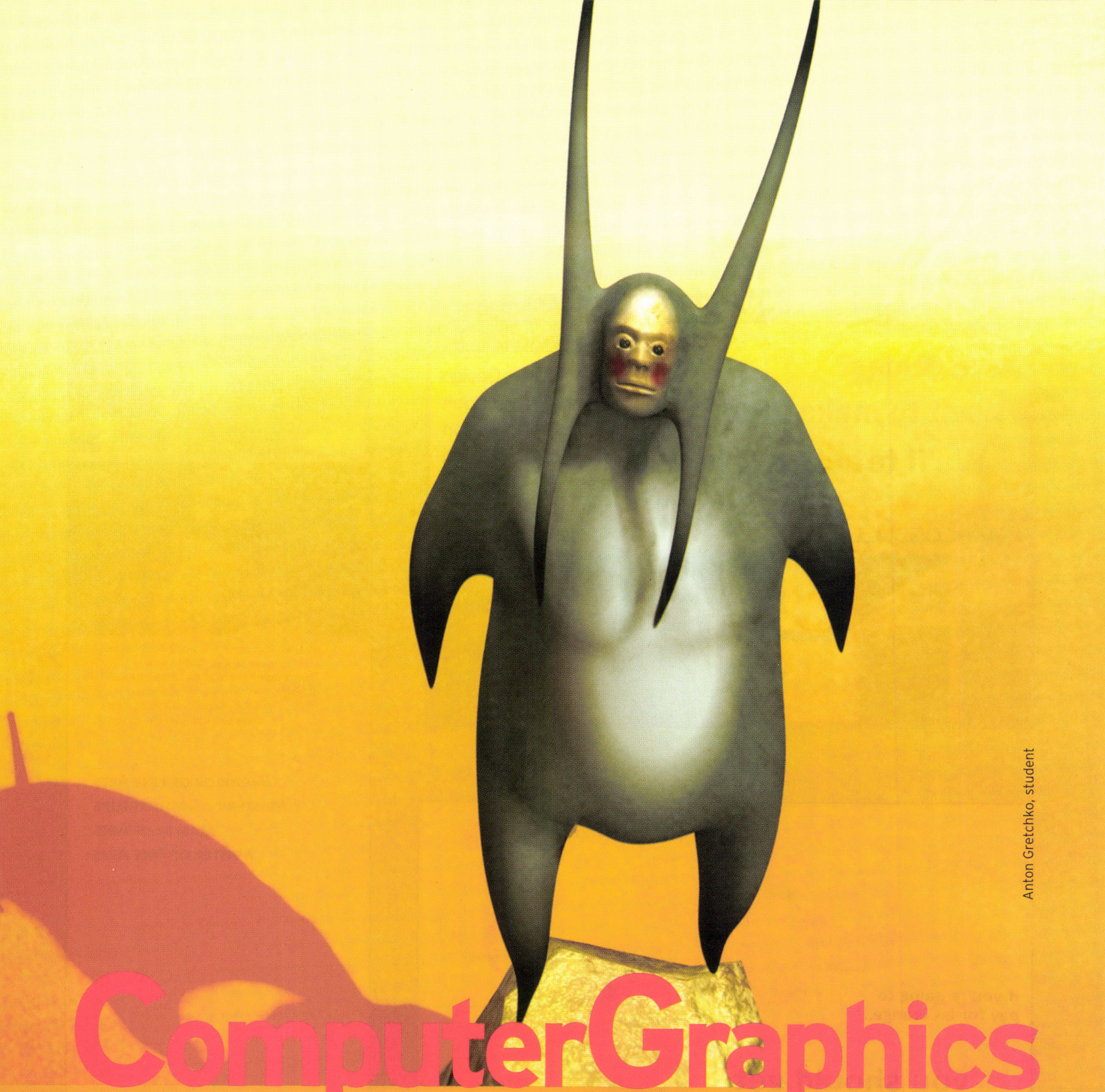


*It's Alive!*, Alias/Wavefront Maya,™ Adobe Photoshop,® Deep Paint 3D™ and Adobe After Effects®
**Terry Ziegelman**, St. Louis, Mo., second year M.F.A.;
**Paul George**, Valencia, Venezuela, second year M.F.A.;
**Stephen Johnson**, Charleston, S.C., B.F.A.;
**Jamie Kirschenbaum**, Reston, Va., B.F.A.

BACHELOR OF FINE ARTS
MASTER OF ARCHITECTURE
MASTER OF ARTS
MASTER OF FINE ARTS

Architectural History
Architecture
Art History
**Computer Art**
Fashion
Fibers
Furniture Design
Graphic Design
Historic Preservation
Illustration
Interior Design
Media and Performing Arts
Metals and Jewelry
Painting
Photography
Product Design
Sequential Art
Video/Film

AN INTERNATIONAL UNIVERSITY FOR THE ARTS
Savannah, Georgia
800.869.7223 | www.scad.edu



Anton Gretchko, student

# ComputerGraphics

- 2D Animation
- 3D Animation
- 3D Modeling
- Character Design
- Computer Arts
- Computer Illustration
- Digital Imaging
- Digital Media
- Digital Photography
- Gaming
- Visual Effects
- Web Design

New online classes available at www.academyart.edu.

AA | BFA | MFA Degrees | Portfolio Development | Apply now for summer, fall & spring semesters.

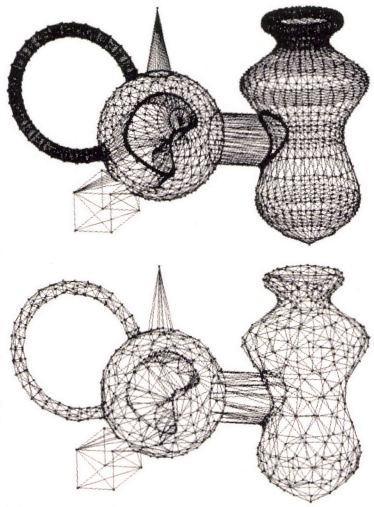
**AcademyOfArtCollege** | 1.800.544.ARTS | www.academyart.edu

79 New Montgomery St. | San Francisco, CA 94105 | Nationally Accredited by ACICS, NASAD & FIDER | Established in 1929

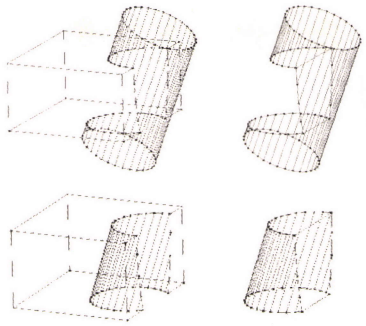
# POPLib 2.0
**Polygon Optimization Library**

POPLib 2.0 is available as C++ Source
Includes: Tessellation; Decimation; Booleans; Sectioning; Ray Firing; and other Polygon and Geometry Tools
Great tools for Software Developers!

## Decimation



## Booleans



www.PolygonLibrary.com
**Click Here for Fewer Polygons!**
**Click Here for Better Polygons!**
**Click Here for Faster Polygons!**

+1.858.538.3800

info@PolygonLibrary.com

## Your project is awesome.

## Demand the best audio.

music • sound • voice

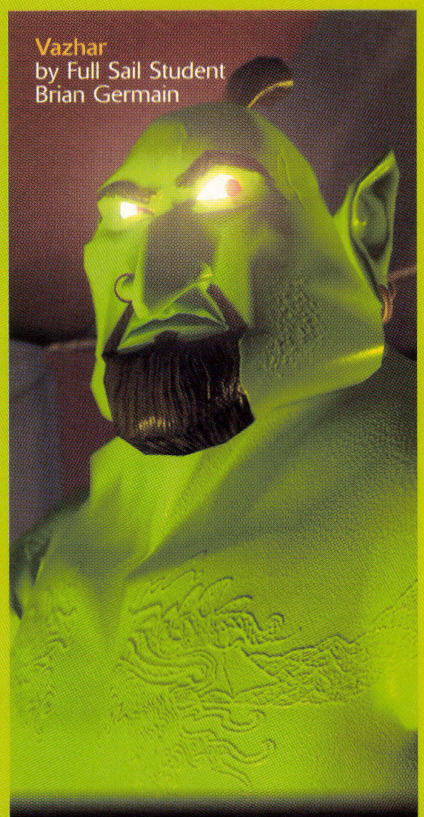
the sound department

**415.454.8988**

www.thesounddepartment.com








FULL SAIL
Real World Education

**School of:**
**Game Design**
**Computer Animation**
**Digital Media**
**Film**
**Audio**
**Show Production**

**800.226.7625**
**www.fullsail.com**

3300 University Boulevard
Winter Park, FL 32792

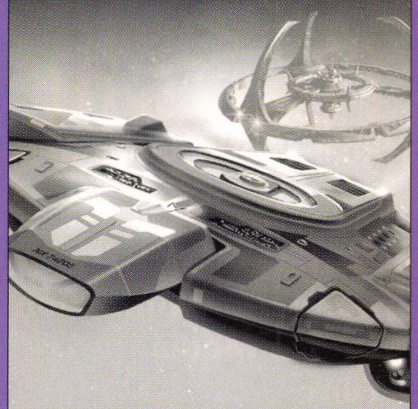
- Financial aid available to those who qualify.
- Job placement assistance.
- Accredited by ACCSCT.

©2001 Full Sail, Inc. All rights reserved.
The terms "Full Sail," "Full Sail Real World Education," and the Full Sail logo are either registered service marks or service marks of Full Sail, Inc.

# Let Us Take You to the Next Level.

For years we've helped entertainment and toy companies with licensed product packaging and promotions. Take your project to new heights – put our expertise to work for you.

Packaging • Advertising
Copy writing • Branding
Logo and Icon Development
Marketing Strategy & Messaging
CD / DVD / Video Promos & Demos
Web and Email Campaigns
POP Solutions



BRAINDANCE

Intelligent creative.
Measurable results.

(323)550-4590
www.braindance.tv
writeus@braindance.tv

we make models
we make textures
we make animations

we make art

we made art for these:

THE SIMS
THE SIMS - Livin' Large
THE SIMS - House Party
THE SIMS- Hot Date
THE SIMS - Family Vacation
Myth 3 - The Wolf Age
S3 - Unreal Tournament Edition
...and a lot more

NEW PENCIL

80 Liberty Ship Way, Suite 6
Sausalito, CA 94965
phone 415.339.1800
fax 415.339.1803
web www.newpencil.com

·MUSIC
·SPEECH ·SOUND
·AUDIO INTEGRATION

OMNI Interactive Audio

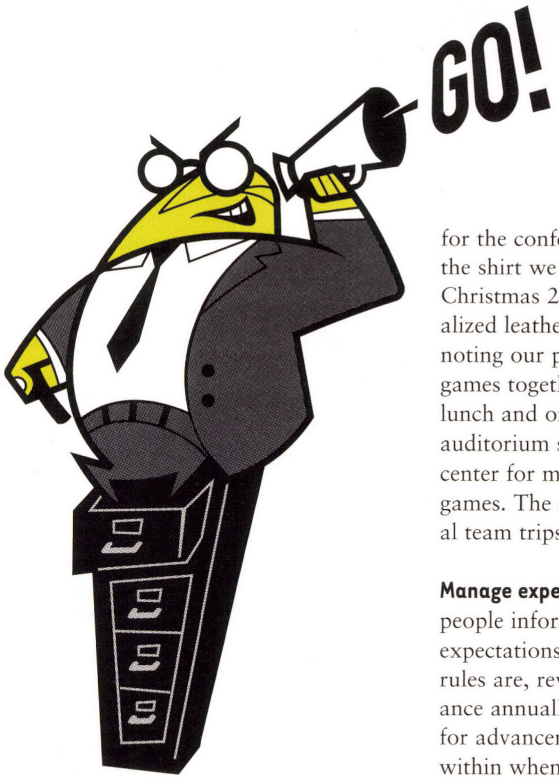
- Veteran team with over 10 years game audio experience each.

- Experience on **PlayStation 2**, **Xbox** and **Gamecube**, online **PC** games, mobile platforms.

- Complete audio package, or support for an overworked in-house audio team.

- Hassle free, efficient audio integration including audio tools.

www.OmniInteractiveAudio.com




*continued from page 56*

ects (which leads to better royalties), offices, and perks. The founding owners of the studio decided wisely that owning a smaller slice of a big pie was better than owning all of a small pie. Our successful employee ownership plan has created incentives for contributors to remain committed for the long term.

**Take pride in our work.** We are the harshest critics of our own games. Our opinion leaders (from anywhere in the company) set a high standard for all crafts. At company meetings or in e-mail threads we acknowledge regularly those people who have done something extraordinary. We encourage and even require all developers to play-test our current game at least once a week. Play-testing creates a very personal connection to the project and gives each person the chance to see his or her work in action. Through play we build respect for what other team members are doing. Play-testing establishes personal ownership of the game and creates inspiration to make it special, in addition to being fundamental to our process of designing by playing.

**Foster camaraderie.** We encourage team spirit through company events such as work-time movie outings (*Lord of the Rings, Final Fantasy,* and the like), social events (company picnics, game launch parties, Christmas parties), in-house game tournaments, and conference trips. At the end of each of our projects, we traditionally divide the company into evenly matched teams for a final tournament, complete with prizes. Everyone in development at ES gets to attend either GDC or E3 each year, if they wish. At such conferences we party as a group. We make a new company shirt each year for the conference season and coordinate the shirt we wear each day. For Christmas 2000, everyone got a personalized leather bomber jacket with badges noting our projects. We play a lot of games together, including board games at lunch and online after hours. Our large auditorium serves as an entertainment center for movie night and sports playoff games. The studio also pays for occasional team trips to sporting events.

*We are trying to create careers at Ensemble Studios, not brief career stops.*

**Manage expectations.** As part of keeping people informed, we try to manage expectations by making clear what our rules are, reviewing individual performance annually, and offering opportunities for advancement. We promote from within whenever possible. During crunch periods we work 10 A.M. to 11 P.M. four nights a week (with Wednesday or Friday off at 6). We try to schedule crunch weeks far in advance so families can be prepared. During crunch, we provide catered lunch and dinner.

**Get it all together.** Striving for high product quality, commercial success, and great team spirit has worked for us so far. We don't see how we could reach one of these goals independent of the others.

**BRUCE SHELLEY** | *Bruce helped found Ensemble Studios in early 1995 and contributed to the design of the* AGE OF EMPIRES *series of games. Prior to joining Ensemble Studios he is best known for assisting Sid Meier with the design of the original editions of* RAILROAD TYCOON *and* CIVILIZATION.

## ADVERTISER INDEX


| COMPANY NAME | PAGE | COMPANY NAME | PAGE | COMPANY NAME | PAGE |
|---|---|---|---|---|---|
| 3DLabs | 13 | Gateway Arts | 21 | Qualcomm | 3 |
| Academy of Art | 52 | IntegrityWare | 53 | RAD Game Tools | C4 |
| Blue Shift | 50 | Intrinsic | 45 | Savannah College of Art and Design | 51 |
| Codeplay | 5 | Konami | 50 | Softimage | C3 |
| Compaq | 25 | Microsoft | 49 | Staccato | 23 |
| Cogswell Polytechnical College | 51 | New Pencil | 54 | Studio Search | 50 |
| From Art to Design | 54 | Numerical Design Ltd. | C2 | The Sound Department | 53 |
| Full Sail Real World Education | 53 | Omni Interactive | 54 | Vancouver Film School | 67 |
| GameSpy | 11 | Perforce | 17 | Wenner Media | 9 |

# Sells Like Team Spirit

Illustration by Paul Howalt

Game development is a very competitive and demanding business, and few new studio start-ups mature to the point where they can make repeatedly commercially successful games. Even ostensibly mature studios may suffer a lot of turnover or even blow apart under the stress.

At Ensemble Studios (ES), we have managed to beat the odds. We think the keys to our success so far are an absolute commitment to making games of the highest quality, crafting games with broad commercial appeal, partnering with (and now being part of) a great publisher (Microsoft), and building a talented and motivated team. There are a number of things ES does to encourage the team spirit necessary to compete in this tough industry. And like any good studio, we are always looking for suggestions on how to improve.

**Put people first.** Studio head Tony Goodman has continually stressed the importance of finding talented people, providing a great studio culture (workplace, equipment, perks), and encouraging positive team spirit. Company morale is discussed at every management meeting. We are trying to create careers at ES, not brief career stops. Our recruiting and interview process is sometimes painful, but we want to add people with a great chance to fit.

**Keep people involved.** We emphasize communication within the studio and have experimented with various ways to improve that. Everyone at ES gets their say about most major decisions the studio makes. We regularly poll the company on questions such as project topics, feature sets, marketing materials, and changes in company policies. Even if the decision cannot be a company vote, everyone has a chance to voice his or her opinion. We hold a company meeting each week and hold impromptu company meetings to share immediate news. We try to minimize rumors and uncertainty.

The small group that manages the studio meets off-site at roughly six-month intervals. Before this meeting, we give everyone in the studio an opportunity to submit items they wish to be discussed and then we brief the company afterwards. In early March 2002, for example, we met to discuss our product plan for the near future, which we then shared with the company.

**Share the success.** We consider large cash project bonuses divisive and counterproductive (and in fact can lead to turnover rather than retention). Instead we decided to share ownership of the company through stock options and reinvest our profits back into funding our own proj-

continued on page 55

"[REAL-TIME SHADERS] ALLOW FOR PREVIEWING OF EFFECTS ON THE XSI VIEWPORT **DURING CONTENT CREATION**, SOMETHING THAT HAS PREVIOUSLY ONLY BEEN POSSIBLE ON **THE XBOX**."
– Yuichi Masuya, Microsoft Co., Ltd., Xbox Division

JSRF – Jet Set Radio Future Original Game ©SEGA ©SEGA/Smilebit 2002

## WHAT YOU SEE IS WHAT YOU GET. ONLY IN **XSI**™ v.**2.0**

Artists and developers, here's the game plan: Start with **SOFTIMAGE|XSI** v.**2.0**, the only true creative choice for sophisticated real-time effects.

**XSI Real-Time Shaders** deliver **state-of-the-art shading** control along with the industry's only native **nonlinear animation (NLA)** system for action-based motion. Now you can work with your art the same way your game engine does.

Feel like calling the shots yourself? XSI v.2.0 internals are fully exposed with the new integrated **Software Development Kit (SDK)** and file I/O is seamless via the free, extensible **dotXSI**™ **file format**.

Ready to play? Visit **www.softimage.com/v2**, order the new XSI Experience CD and **let the games begin**.

2.0

SOFTIMAGE® | XSI™

Innovate. Create. Collaborate.

AMD Athlon MP
AMD Certified

© 2002 Avid Technology, Inc. All rights reserved. Softimage and Avid are registered trademarks and XSI, the XSI Logo, dotXSI and Make, Manage and Move Media are trademarks of Avid Technology, Inc. AMD, the AMD logo, AMD Athlon MP and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other trademarks contained herein are the property of their respective owners.

Avid
Make, Manage and Move Media™

# The Best in Game Development Technology

## GRANNY 3D 2.0

Granny 2.0 combines the **sophisticated dynamic 3D animation system** of its predecessor with a completely new and optimized run-time engine to deliver incredible performance in an extraordinarily small memory footprint perfect for consoles as well as PCs and Macs. Brand new, fully customizable and scriptable **MAX & Maya** plug-ins export all material and shader graph data, multiple texture and color channels for vertices, complete skinning information, and user-defined node attributes and scripts, all to a new, completely open, extensible, and easy-to-use file format.

With all this, plus utilities for advanced camera control, collision detection, animation and texture processing, MIP-map generation, and **Bink and S3TC** texture compression, Granny adds years worth of features to your 3D engine in just a few hours of integration time!

## BINK VIDEO TECHNOLOGY

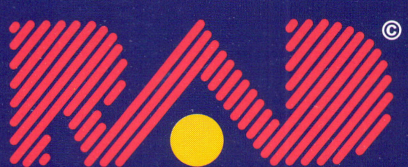
The very best video codec - make your videos shine! Unrivaled quality (**better than DVD** and MPEG II). Up to **three times faster** than other true-color codecs. Perceptibly lossless, **8 to 1 audio compression**. Decompresses to DirectDraw, DIBSections, YUV overlays, or any other memory. Available for Win32, MacOS, Xbox, and now **Nintendo GameCube**. Think Bink!

## SMACKER VIDEO TECHNOLOGY

The best 256-color codec on the planet! Smacker is the codec of choice for **256-color games** (of course), games targeting Pentium 133s and below, games with video sprites or video transparency (**much faster in 256 colors**), or games with extremely high-resolution videos (800x600 and up). Smacker is available on Win32, Win16, DOS, and MacOS.

## MILES SOUND SYSTEM

The ultimate sound system for PC and Mac! Miles supports 2D and 3D audio, MIDI with DLS, streaming, CD audio, **DSP filtering**, **MP3 playback (patent rights included)**, **Internet voice chat** (2900, 2400, and 1200 bps codecs), Creative **EAX 1, 2 & 3**, Aureal **A3D 1 & 2**, RSX 3D, DirectX 7, Dolby Surround, QSound, **super-fast software EAX emulation**, and more!




**425 893 4300**

**www.radgametools.com**

**POWERFUL TECHNOLOGY. EASY TO USE. NO ROYALTIES.**